Rochester Institute of Technology
RIT Scholar Works

Theses | Thesis/Dissertation Collections

11-2016

Detection in Aerial Images Using Spatial Transformer Networks

Daniel Chianucci
ddc1284@rit.edu

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Chianucci, Daniel, "Detection in Aerial Images Using Spatial Transformer Networks" (2016). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Detection in Aerial Images Using Spatial Transformer Networks

DANIEL CHIANUCCI

Detection in Aerial Images Using Spatial Transformer Networks

DANIEL CHIANUCCI

November 2016

A Thesis Submitted
In Partial Fulfillment
Of the Requirement for the Degree of
Master of Science
In
Computer Engineering

Department of Computer Engineering

Detection in Aerial Images Using Spatial Transformer Networks

DANIEL CHIANUCCI

Committee Approval:

Date:_____

Dr. Andreas Savakis
Primary Advisor – R.I.T. Dept. of Computer Engineering

Date:_____

Dr. John Kerekes
Committee Member – R.I.T. Center for Imaging Science

Date:_____

Dr. Dhireesha Kudithipudi
Committee Member – R.I.T. Dept. of Computer Engineering

I would like to dedicate this thesis to my family and friends who helped me get to the point where I am today.

Acknowledgments

I would like to thank my advisor Dr. Savakis, for his advice and guidance during the development and experimentation of this thesis. I also would like to thank my committee members Dr. Kerekes and Dr. Kudithipudi. Finally I like to thank my peers in the Computer Vision Lab Peter Muller, Bret Minnehan, and Sriram Kumar who provided both insight and entertainment during the long hours at the lab.

Abstract

Many tasks in the field of computer vision rely on an underlying change detection algorithm in images or video sequences. Although much research has focused on change detection in consumer images, there is little work related to change detection on aerial imagery, where individual images are recorded from aerial platforms over time.

This thesis presents two deep learning approaches for detection in aerial images. Both systems leverage Spatial Transformer Networks (STN) that identify the coordinate transformation for their localization capabilities. The first approach is based on a semi-supervised approach which learns to locate changes within a difference image. The second is a fully-supervised approach which learns to locate and discriminate relevant targets. The supervised approach is shown to locate nearly 78% of positive samples with an Intersection Over Union (IOU) criterion of over 0.5, and nearly 94% of positive samples with an IOU over 0.3.

Table of Contents

Dedication ii
Acknowledgments iii
Abstract iv
Table of Contents v
List of Figures vii
List of Tables viii
Chapter 1. Introduction 1
1.1 Motivation 1
1.2 Thesis Contribution 2
1.3 Document Structure 3
Chapter 2. Background 5
2.1 Change Detection in Consumer Data 5
2.2 Change Detection in Aerial Images 7
2.3 Deep Learning 11
2.4 Spatial Transformer Networks 14
Chapter 3. Preliminary Experiments 18
3.1 Difference of Reconstructions 18
3.2 Difference of Encodings 19
3.3 Temporal Feedback 21
Chapter 4. Semi-Supervised Approach 23
4.1 Methodology 23
4.1.1 Network Architecture 23
4.1.2 Network Training 25
4.1.3 Sliding Window and Post Processing 26
4.2 Results 28
4.2.1 Dataset 29
4.2.2 Localization Results 30

4.2.3 Energy Thresholding 34
4.2.4 System Results 37
4.3 Discussion 39
4.3.1 Fixed Scale Parameter 39
4.3.2 Dataset Complexity and Overfitting 41
4.3.3 Semi-Supervised vs Unsupervised 41
4.3.4 System Window Overlap 42
4.3.5 Real World Viability 42
Chapter 5. Fully Supervised Approach 43
5.1 Methodology 43
5.1.1 Network Architecture 44
5.1.2 Training 46
5.1.3 Post processing 47
5.2 Supervised Results 48
5.2.1 WPAFB Dataset 48
5.2.2 Localization Results 50
5.2.3 Network Weights 55
5.2.4 Detection Results 59
5.2.5 System Results 63
5.3 Discussion 67
5.3.1 Fixed Scale vs. Free Scale 67
5.3.2 Low Resolution Targets 68
5.3.3 System Performance 68
Chapter 6. Conclusions 70
6.1 Concluding Remarks 70
6.2 Future Works 71
Bibliography 73
Appendix I – Evaluation Equations 77

List of Figures

Figure 1: Activation Functions 13
Figure 2: Application of a Spatial Transformer Sampling Grid 15
Figure 3: Autoencoder Difference of Encodings. 20
Figure 4: Temporal Feedback Mechanism. 21
Figure 5: Semi-supervised Network Architecture. 24
Figure 6: AICD Scene 94 - Viewpoint 0 With Changed Area Circled 31
Figure 7: Localization Network IOU Example - 32
Figure 8: Qualitative Unsupervised Network Predictions 33
Figure 9: Prediction Truth Scenarios 34
Figure 10: Unsupervised Energy Based Detection ROC 35
Figure 11: Semi-Supervised Energy-Based Detection Precision vs. Recall 36
Figure 12: Semi-Supervised System Miss Rate vs FPPI 39
Figure 13: Semi-Supervised System Example Predictions 40
Figure 14: Supervised Network Architecture 45
Figure 15: Full WPAFB Dataset Scene 49
Figure 16: Localization and Detection Examples for Free-Scale Network 51
Figure 17: Localization and Detection Examples for Fixed-Scale Network 52
Figure 18: Correct Detections with Low IOU 55
Figure 19: First Convolution Layer Weights 56
Figure 20: First Dense Localization Layer Weights 57
Figure 21: First Dense Layer Detector Weights 58
Figure 22: Training Loss History 59
Figure 23: Supervised Detector Receiver Operating Characteristics 60
Figure 24: Precision vs Recall Curve 62
Figure 25: Supervised System Log-Log Miss Rate vs. FPPI on the First K-fold split 64
Figure 26: Supervised System Predictions Using Detection Threshold $\alpha = 0.8$ 65
Figure 27: Supervised System Predictions On Second Holdout Image 66

List of Tables

Table 1: Semi-supervised Localization Network Average IOU Results 33
Table 2: Unsupervised Area Under Curve and mean Average Precision 36
Table 3: Detailed Energy Based Detection Characteristics (min IOU=0.5) 37
Table 4: Fixed Scale Network Average IOU Localization Results 53
Table 5: Free-Scale Network Average IOU Localization Results 53
Table 6: Fixed Scale Network K-Fold Localization Results 54
Table 7: Supervised Detector ROC Areas 60
Table 8: Detailed Network Test Results For Split 1 (min IOU=0.5) 61
Table 9: Supervised Network Mean Average Precision 62

Chapter 1. Introduction

Change detection, as the name suggests, is the task of finding differences between a set of images or across video frames. Many tasks in the field of computer vision rely on an effective underlying change detection algorithm. For instance, motion detection and object tracking rely on a change detection algorithm to determine where to focus attention. Surveillance systems such as Closed-circuit Television (CCTV), and Unmanned Aerial Systems (UAS) generate huge amounts of data. Without automated change detection, the task of analyzing each image frame would become unmanageable.

Although change detection seems like a simple challenge, an algorithm can be easily confused by many sources of background noise. Illumination differences, camera movement, animated backgrounds, and even compression artifacts can all affect an algorithms performance. Simple methods, such as difference thresholding can be used [1]; however such methods generally perform poorly in all but the most ideal scenarios. In order to robustly detect changes in real world images, more advanced methods must be employed.

1.1 Motivation

Exploitation of aerial images, especially Wide Area Motion Imagery (WAMI) is important for military applications, as can be seen by the number of related challenges set by the United States Air Force Research Labs (AFRL) [2]. In fact, the AFRL listed Wide Area Coverage as an integral aspect of its Intelligence, surveillance and reconnaissance

doctrine [3]. With the changing nature of interactions between adversaries, it is important that armed and security forces are able to keep an eye on large theaters, while at the same time having the ability to focus on smaller areas of interest.

Many studies have focused on finding a robust algorithm for change detection in consumer imagery. Benchmark datasets and leaderboards such as ChangeDetection.net (CDNET) [2] ease the process of creating and evaluating new algorithms, and also help immensely with making comparisons to other's results. However, these benchmarks and datasets aren't applicable to wide area coverage. While current datasets focus on high resolution images where the main topic is the target in question, wide area motion images are very low resolution, and interesting objects fill only a minute portion of the scene.

Wide Area generally means that the images cover an area greater than 50 square miles and so it is infeasible for a human analyst to reliably extract the needed information from a stream of such images. Given the sheer size of WAMI images, the number of pixels is unmanageable for algorithms developed for consumer applications. This fact has led to the research and development of techniques adapted towards image processing in WAMI data. Besides the number of pixels, such research has to deal with other challenges presented by WAMI data. Small object signatures, as well as low object signal to noise ratios (SNRs) mean that simple detection algorithms are not sufficient.

1.2 Thesis Contribution

The primary contribution of this thesis is the introduction of two aerial change detection techniques using a spatial transformer network. Spatial transformers are a type of neural

network that are able to learn how to parameterize an image transformation which is beneficial to a given training objective. They have been shown to excel in fine grained classification tasks [4][5], but have not been used in the context of change detection or explicit localization. Rather than using the spatial transformer to find an area which best differentiates class, it is used to learn the bounding box of a target.

Two approaches are explored; the first is a semi-supervised change detection method which attempts to find an area of maximal change in a difference image. Given that no explicit labels are used for training, this method is unable to determine the relevance of a change, and instead relies on heuristics to reject false positives. A second fully supervised methodology is employed in order to allow the network to learn the structure of specific targets. Rather than finding changes directly from a difference image, this methodology requires a set of registered images and compares the predictions from each. Both networks are used in a sliding window based approach in order to make multiple detections per image.

1.3 Document Structure

Following this introduction chapter, Chapter 2 presents a review of previous works. First, an overview of the state of change detection in consumer data is given. This field has a number of standardized benchmarks and datasets, and it is relatively straightforward to compare methodologies. Next, an overview of change detection techniques for aerial images is presented. This field of change detection is much more diverse than that of consumer data, and there is no standard benchmark for comparisons. This is followed by a section devoted to deep learning architectures and training methods.

Finally, the chapter concludes with an overview of Spatial Transformer Networks which are used extensively in this thesis. Chapter 3 provides a narrative of experimental work which was done prior to the final focus of change detection in aerial imagery. These experiments were focused on finding a method of robust change detection in consumer data using autoencoder features. When these features didn't perform as well as expected, they provided valuable lessons as the focus was changed to spatial transformer networks and aerial imagery. Moving forward, Chapter 4 goes into detail about the semi-supervised aerial change detection method that was explored. First the methodology is described, including the neural network architecture, the training procedure, and the post-processing steps. Results are presented, with an overview of the dataset, as well as qualitative and quantitative test outcomes. Finally, a discussion of interesting phenomena, shortcomings, and challenges is given. Chapter 5 follows the same structure as Chapter 4, but describes methods, evaluation, results and discussion for the fully supervised methodology. Finally, Chapter 6 gives concluding remarks, and a statement about possible future works.

Chapter 2. Background

Change detection is fundamental to many computer vision tasks such as tracking, medical diagnosis, and surveillance. Although there have been many studies which introduce new and efficient methods of change detection, most are designed to work with full motion video taken from ground level cameras. These sequences tend to have a very high spatial resolution, and so do not directly correlate with those taken from an aerial source.

Although research for this thesis began in the field of change detection in consumer data sequences, it eventually transitioned to the task of finding changes between a set of two still images taken from an aerial vantage point. Aerial images present a large number of challenges including camera motion, large shadows, and extremely small targets to name a few. The data driven method of deep learning will be used to train a neural network to locate change targets within large aerial images.

2.1 Change Detection in Consumer Data

Change detection in consumer data is a well-researched topic, and as such has the benefit of associated standardized benchmarks and datasets. This standardization made the field an ideal starting point for the research in this thesis.

A broad overview of multiple change detection algorithms is provided in [1]. The simplest algorithms consist of thresholding a difference image. A difference image as a map of pixel differences generated using a pixel-wise distance measure between two images. Due to their simplicity, these types of approaches are extremely fast, however

they suffer from poor performance in real world images. These simple methods are unable to differentiate between relevant changes, such as those caused by a moving person, or from irrelevant changes such as those caused by a rustling tree or shadows.

This lack of performance in real life scenarios motivated the development of more robust methodologies, however, without a standard dataset, these methods were difficult to compare. As such, the changedetection.net (CDNET) dataset was created in order to simplify the process of benchmarking the latest advances in change detection [6].

Most approaches submitted for the CDNET dataset ranking build a background model which is used as a reference for change detection. A popular model choice is to use one or more probability distributions to represent the likelihood that a pixel value will occur in an area. However, simple distributions such as those used in Gaussian Mixture Models [7] and Region-based Mixture of Gaussians [8] do not perform as well as newer methods.

The next step in improving distribution based methods was to utilize models for both background and foreground. Methods such as Flux Tensor with Split Gaussian Model (FTSG) [9] and Sharable Model [10] generate such models and take temporal, in addition to spatial, regions into account. These additions place the two methods towards the top of the CDNET submission rankings, however higher performing methods do exist.

The two current state of the art methodologies are Pixel-based Adaptive Word Consensus Segmenter (PAWCS) [11] and Self-Balanced Sensitivity Segmenter (SuBSENSE) [12]. These methods are extremely similar in that they do away with the Gaussian models and replace them with a set of background frames composed of Local Binary Similarity

Pattern (LBSP) descriptors. A feedback loop is used to adjust both the background model as well as the comparison threshold according to a stability metric attributed to each pixel. Stochastic updates are incorporated to prevent the model from becoming stale.

Finally, the top ranked method, termed In Unity There Is Strength or IUTIS, is an aggregate of other submitted methods [13]. It uses genetic programming to learn a fusion strategy to aggregate other methodologies' results.

One common aspect the majority of the approaches submitted to CDNET share is that they use engineered features and statistical methods.

2.2 Change Detection in Aerial Images

Due to a lack of a standardized dataset and challenges, the work done in the field of aerial image change detection is less focused on benchmark data. The large number of image modalities, scales, and spectral characteristics makes it difficult to compare approaches. In addition to simple RGB images which is the dominant mode for consumer data, aerial data also comes in forms such as Wide Area Motion Imagery (WAMI), Synthetic Aperture Radar (SAR), Panchromatic, and Multispectral. Each of these modalities comes with its own set of benefits and challenges which must be dealt with. That being said, this section presents a selection of works which directly relate to change detection in aerial images independent of the type of data.

Change detection from aerial images started in the 1960's when Rosenfeld proposed a variety of correlation metrics to measure the similarity between image features [14]. Early research into change detection came to the realization that not only do geometric and radiometric distortions need to be accounted for, but also that changes would need to be categorized to reject irrelevant variation such as those caused by clouds or shadows.

These challenges led to the discovery of symbolic techniques, in the 1970's and 1980's, which emphasized the detection of changes based on the shapes, sizes, and radiometric properties contained within the images [15]–[17]. These techniques generally use a segmentation technique to separate the image into discrete areas, which can then be classified based on shape, texture, or spectral features. In particular, [18] uses edge detection, and Hough line transforms to detect changes based on straight line matching. This approach is particularly well suited for updating urban Geographic Information System (GIS) data due to the often grid like city plans.

With advancements in remote sensing technologies and the introduction of Wide Area Surveillance (WAS) new techniques were developed for detecting man made changes in in larger scenes. In [19], a man made change detection technique is described which involves modeling the spectral and size changes which can be expected during facility construction. These changes are harder to model than naturally occurring changes due to the fact that they aren't as predictable as changes observed for crops and other vegetation over time.

In the early 2000's an uptick in research involving Artificial Neural Networks (ANN) is seen. However due to limitations in both computing power and neural network

techniques, these studies use very small neural networks consisting of only a few layers, each using less than 15 neuron units. In [20] Principal Component Analysis (PCA) was used to generate features from multispectral pixel data. These features were then used to train and evaluate a three layer ANN to classify features into land use classes. These predicted classes were then used to determine if a change had occurred in land use between two images. Similarly, [21] uses a three layer neural network to predict an after image from a before image, and vice versa. A set of heuristics is then used to determine if the prediction is unusual.

Approaching the late 2000's, a type of neural network called a Self-Organizing Feature Map (SOFM) began to be used. An SOFM is a nonlinear generalization of PCA [22], and is used to learn a low dimensional representation of the input space in an unsupervised manner. Both [23] and [24] use an SOFM network to classify areas into changed and not changed regions, however where the first classifies entire segments at a time, the second classifies individual pixels.

A more recent method which uses an SOFM was introduced in [25]. This method detects changes to building structures in high resolution RGB images. First, a histogram of illumination invariant features is used to train an SVM for building detection. Then, an SOFM based active contour model is used for boundary extraction. The extracted boundaries can then be used to compare areas between images.

A non-machine-learning approach was introduced in [26], which uses optical flow to detect changes between a set of aerial images. This paper introduced the simulated Aerial Image Change Detection dataset, which is also used in this thesis. They perform

optical flow on a set of registered images in order to remove changes caused by simple parallax effects.

A statistical tile based methods was introduced in [27], which split an image into equally sized tiles and performed change detection at the tile level rather than the pixel level. This approach takes tiles from a set of registered images and determines a tiles change potential based on a set of complimentary metrics. The difference of spectral distribution means, or mean-shift, is used to quantify large scale changes within a tile, while the greatest distance between a pixel and the distribution mean, or and outlier-distance, is used to quantify small scale changes. Combining these metrics allows tiles which contain large or small scale changes to be detected.

Similarly, [28] introduces a graph theoretic approach to detect tile level changes between two registered images. With this approach, a weighted adjacency matrix is calculated for each tile, along with the Standard Deviation of Edge Weights edges (SDEL). A metric coined the Normalized Edge Volume (NEV) is used to measure the spectral variability between two registered tiles. Large scale changes within a tile correspond with large NEV values, whereas small scale changes correspond with large SDEL value. Although both tile based methods were shown to work well in variety of situations, the resolution of the output change map is limited by the size of the tiles.

Deep learning has been used recently to detect changes between a set of SAR images. In [29], a two input deep neural network is trained to produce a change map directly. This allows the network to learn relevant features, without the need for computing a difference

image. This method however was only shown to work well for large change targets such as coastlines, and farmland fields.

Finally, recent research by Kitware and AFRL [30], has found that change detection in WAMI images can be greatly enhanced by applying the differencing to a detection response map rather than to the raw image pixels. By running a Histogram of Oriented Gradients (HOG) based vehicle detection Support Vector Machine (SVM), this paper is able to create a heat map of sort denoting the likelihood that a vehicle is present. When two of these maps are registered and differenced, the pixels with large differences are taken to be potential changes. This process significantly reduces the number of residual bright areas caused by illumination and parallax differences. This method of detecting changes between detection maps rather than raw images was a motivation for the supervised detection methodology presented in this thesis.

2.3 Deep Learning

Deep learning is a term which encompasses neural networks containing many layers. These networks aim to emulate the connections in the human brain in order to learn a rich and robust set of features. Deep learning approaches have been successfully used in many machine learning applications, especially in the area of computer vision. Many variants of deep networks exist, each with its own benefits and detractors.

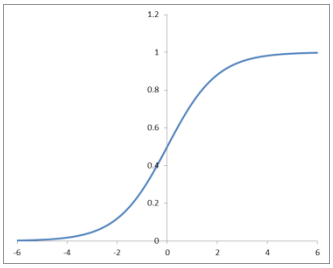
The basic concept of a neural network can be represented by a graph of compute nodes. Each node takes a set of inputs and provides a single output. Although individually, each node is incapable of modelling complexity, together they are able to approximate any

continuous function [31]. This sounds like an end all solution; however the difficulty comes in training the network to perform as desired.

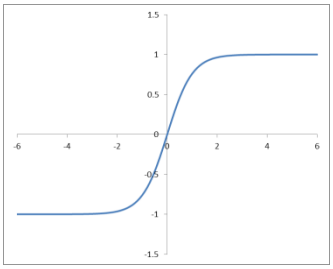
Training a neural network consists of multiple steps. First, network architecture must be developed, or borrowed. Basic architectures consist of an input layer, a series of internal hidden layers and finally an output layer. Each layer is generally followed by a nonlinear function called nonlinearity or an activation function. Early deep neural networks consisted solely of dense layers which perform a dot-product of the weights and the input [32]. Common activation functions included sigmoid and hyperbolic tangent functions. However, since each neuron in a dense layer connects to each and every neuron in the previous layer, the number of connection weights which must be learned increases at a rapid pace. This motivated the introduction of convolutional layers in 1998 which allow weights to be shared across spatial extents [33]. Convolutional networks have been shown to excel in many computer vision tasks due to their ability to preserve the 2D structure of an image [34]–[36]. Recently, activation functions being used have shifted towards Rectified Linear Units (ReLU) rather than sigmoidal forms. These nonlinearities take the form of a simple max operation as seen in Figure 1, and don't suffer from issues such as vanishing or exploding gradients seen in other nonlinearities. One issue they do have however is that non active units contribute nothing to the gradient, and as such become permanently inactive. A simple solution is to introduce a small slope for negative samples to make a "Leaky ReLU".

Once the architecture has been selected, the feature learning can begin. A sample is fed to the input layer, and the connection weights are used to propagate the sample through

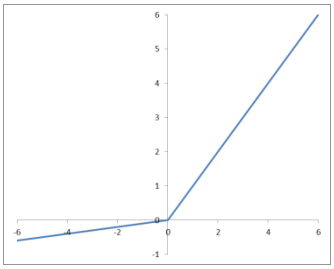
the network onto the output. An objective function such as mean square error or categorical crossentropy then calculates an error metric between the prediction and the ground truth. This error is then passed back through the network as a series of gradients in a process called back propagation in order to update the weights.



$$Sigmoid = \frac{1}{1 + e^{-x}}$$



$$tanh$$



$$Leaky\ ReLU = \max(m \cdot x, x)$$

Figure 1: Activation Functions

Although Neural Networks have proven to be very successful they require a large number of labeled samples to train. One network architecture which aims to solve this is the autoencoder.

In the simplest form, an autoencoder consists of a neural network appended with its own inverse. The network is then trained to reconstruct the input. In essence, an autoencoder is a neural network which aims to approximate the identity function. Because the identity function isn't very interesting in and of itself, constraints are imposed on the network in order to force it to learn interesting structures. For instance, common constraints include making the middle layer much smaller than the input, or enforcing sparsity on the middle layer [37]. This forces the network to learn a compressed representation of the input

space. As with self-organizing feature maps, autoencoders can be thought of as a nonlinear generalization of PCA [38]. A second approach termed a denoising autoencoder involves introducing noise to the input, and train the network to reproduce a clean output [39]. This forces the autoencoder to learn only salient features in the presence of noise.

2.4 Spatial Transformer Networks

A new class of neural networks deemed Spatial Transformer Networks (STN) has recently been introduced [4]. As the name suggests, these networks learn to parameterize a spatial transformation on an image or feature map in order to focus on pertinent areas.

The main building block of a spatial transformer network is the Spatial Transformer layer. This specialized layer does not contain any weights, and as such does not learn on its own. Rather it applies a differentiable transformation given by a parameterized input onto an input feature map. Because the transformation is differentiable, gradients are able to flow through this layer during backpropagation in order to update the weights of both the transformation parameter input, as well as the feature map input. If the transformation parameters are provided by a neural network, as intended, then the backpropagation allows the transformation to be learned.

Although the transformation may take any form as long as it remains differentiable, the authors focus on pointwise affine transformations, given by the form shown in (1), where (x_i, y_i) is a normalized coordinate on the input space, (x_i^t, y_i^t) is a normailized coordinate on the output space, and T_θ is an affine transformation matrix.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = T_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \quad \textbf{(1)}$$

By applying such a transformation to a regular grid $G = \{(x_i^t, y_i^t)\}$, consisting of all target pixel coordinates, an output feature map consisting of input space sampling points is generated. The input image is then sampled and interpolated using the generated points, resulting in a transformed image as shown in Figure 2.

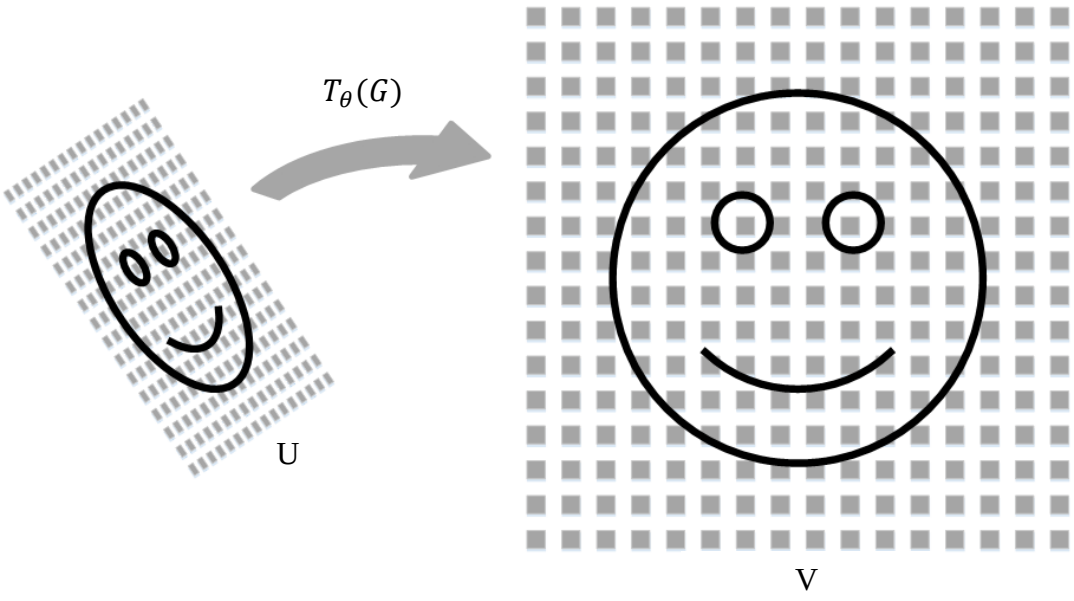


Figure 2: Application of a Spatial Transformer Sampling Grid

Each coordinate (x_i, y_i) generated by the sampling grid represents the location in the source image, U, where a sampling kernel is applied to get the pixel value for the output, V. A bilinear sampling kernel is shown in (2), where V_i^c is the output value for pixel i on

channel c. Similarly, U_{nm}^{c} denotes the input pixel at coordinate (n, m). The two max functions determine the relative weight for each pixel to obtain the bilinear interpolation.

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \cdot \max(0,1 - |x_i - m|) \cdot \max(0,1 - |y_i - n|) \ \forall i \forall c \quad \textbf{(2)}$$

For back propagation to occur, partial derivatives must be calculated. For the bilinear kernel shown in (2), the partial derivatives can be calculated as follows.

$$\frac{\delta V_i^c}{\delta U_{nm}^c} = \sum_n^H \sum_m^W \max(0,1 - |x_i - m|) \cdot \max(0,1 - |y_i - n|) \quad \textbf{(3)}$$

$$\frac{\delta V_i^c}{\delta x_i} = \sum_n^H \sum_m^W U_{nm}^c \cdot \max(0,1 - |y_i - n|) \cdot sign(\max(0,1 - |x_i - m|)) \quad \textbf{(4)}$$

The partial derivative shown in (3), allows gradients to flow backwards from the output to the input feature map. The derivative shown in (4) on the other hand allows the gradients to flow from the output to the sampling coordinates, and in turn the transformation parameters and localization network. Note that the partial derivative with respect to y_i follows a similar equation to (4).

Spatial transformer networks have been shown to perform extremely well in classification tasks [4], [5]. In these tasks the localization portion of the STN will inherently learn to focus onto the portion of the feature map which is best able to discriminate between classes. The ability to learn the size and shape of the important image regions would seem to be a perfect fit for the task of changed target detection; however few if any works have studied this.

With that being said, this thesis proposes a new method to train an STN in order to parameterize the bounding box of changed targets within aerial images. First, a semi-supervised approach to finding changes within a difference image is presented. This method receives no information on the structure of relevant targets during training. In order to better predict only relevant targets, a supervised approach is also introduced. This method integrates a detection network with the semi-supervised architecture which allows structure to be learned.

Chapter 3. Preliminary Experiments

Although this thesis eventually settled on change detection in aerial images, the initial experiments that were run were more geared towards detecting changes in full motion consumer videos. This section serves as a narrative of the process that was followed in order to end at the new focus.

The desired approach was leveraging deep learning features. Because learned features are robust to various sources of noise, it follows that they would excel in tasks, such as change detection, where environmental noise is expected. However, because real world change detection scenarios don't have human labelled ground truths, deep learning is severely limited. This presents an opportunity for unsupervised learning with autoencoders.

3.1 Difference of Reconstructions

The first approach considered was to train an autoencoder, and use its reconstructions to generate a difference image. The thought was that since autoencoders tend to retain only major aspects of an input, differencing the reconstruction rather than the original images would result in a difference image with only major changes. In the case of change detection, where one does not need nor want to detect inconsequential changes, this loss of small details could be a benefit.

Multiple models of autoencoder were tested by trial and error. The final model consisted of three convolution layers using ReLU nonlinearities each followed by a 2x2 max

pooling layer. The first convolutional layer consisted of 32 7x7 filters, the second was 16 5x5 filters, and the final convolution was 16 3x3 filters. The decoder portion of the network was a linear inverse of the encoder described.

A deep convolutional autoencoder model was trained on 32x32 patches from the CDNET video sequences. The training objective was set to minimize the mean square error between the input and output images, the reconstructions still contained remnants of the inconsequential changes. If the network was instead trained to reconstruct a clean background, defined as the pixel wise median of the previous temporal window, the network would simply overfit. The bias of the final layer learned to emulate the mean of the entire sequence, while the weights stop contributing to the output. If the bias was then constrained, the network simply reverts to modeling the input image including irrelevant changes.

3.2 Difference of Encodings

It was then hypothesized that if the difference was taken at the feature level, then the resulting encoding would decode back into a clean difference image similar to what is seen in Figure 3. First an autoencoder was trained using the same method as the "Difference of Reconstructions" autoencoder described above. The encoder and decoder portions were then split apart. A changed image and a clean image were fed through the encoder portion to generate two encodings represented as h and h'. A function, F, is applied to h and h' which generates a third encoding. This third encoding is sent through the decoder to generate a full sized difference image.

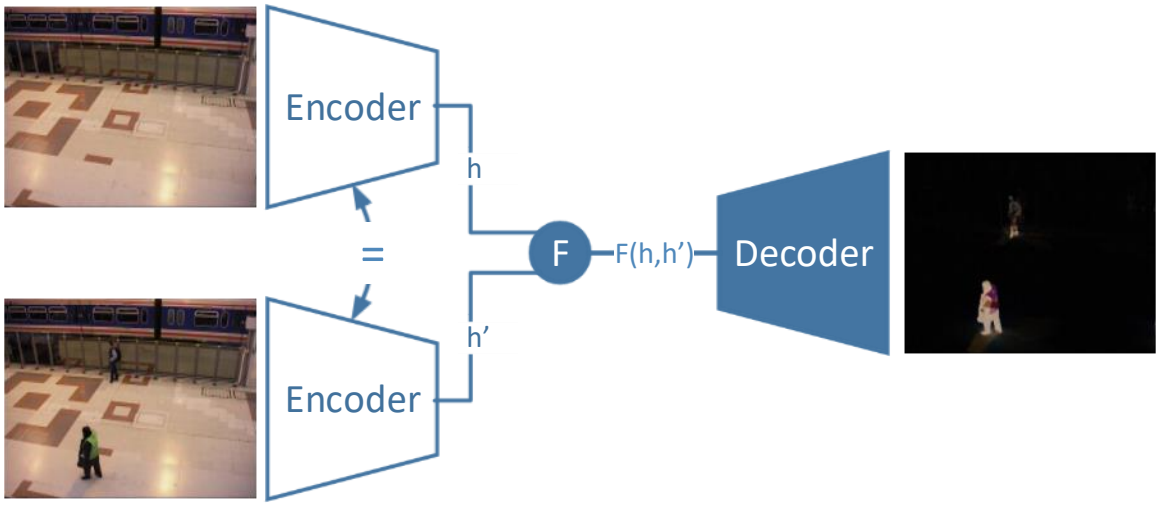


Figure 3: Autoencoder Difference of Encodings.
Both clean and changed images are fed through the encoder portion of the autoencoder. A function F is applied to the hidden representations h and h'. The result is decoded using the decoder portion of the network, thus resulting in a difference image

Because the encoded features are an abstract representation, there is no established method to compare two encodings. Therefore, in an attempt to generate a clean difference image, both the difference and absolute difference were tested. However, the resulting reconstructions tended to be blurry and contain bright halos. These reconstructions were thresholded and cleaned using morphological operations; however evaluation proved that similar results could be obtained by simply applying the same morphological operations to a standard difference image. This finding ultimately led to seeking alternate approaches.

3.3 Temporal Feedback

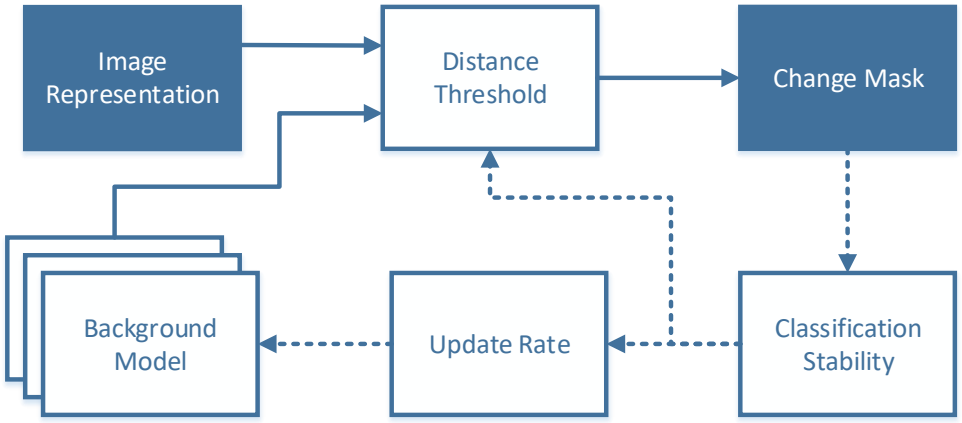


Figure 4: Temporal Feedback Mechanism.
At each time step the autoencoder hidden representation of an image is input, and a change mask is output. A stability metric is used to update the distance thresholds, and background update rate.

Autoencoder features were next considered in a different setting. Looking into the feedback based approaches of SuBSENSE and PAWCS [11], [12], it can be seen that the results are dependent on the robustness of the underlying features being used. A similar feedback mechanism shown in Figure 4 was developed to test the feasibility of autoencoder features in comparison to the LBSP features used in [11] and [12].

Although autoencoder features showed promise in theory, in practice they did not work as well as expected. The LBSP features and feedback used by PAWCS and SuBSENSE produced cleaner results, and were much less computationally expensive. Where LBSP descriptors can be calculated using simple comparison operations, autoencoder features employ the use of multiple convolutions. After the features are computed, LBSP

descriptors are still more efficient in the fact that distance comparison requires only XOR operations, whereas floating point math is required with autoencoder features.

Because of the computational load accompanying autoencoder features, even if detection performance was on par with the state of the art, the low computational efficiency would rule them undesirable for this application. Although there were high expectations for the use of autoencoders in change detection, they tended to not perform as well as desired. For this reason, the focus was altered towards using spatial transformers and aerial images.

Chapter 4. Semi-Supervised Approach

In this chapter a semi-supervised approach to change detection in aerial images is introduced. This method aims to find unstructured changes between two images. This is accomplished using a Spatial Transformer, which is trained to find an area of maximal change in a difference image.

4.1 Methodology

A Spatial Transformer Network was utilized to localize changes transforming its input image so the output viewport contained maximal change. In this case, maximum change is defined as maximum Euclidean distance between the original and changed images. The assumption that significant changes will result in bright areas in the difference image was made, however this heuristic doesn't hold in all cases. It is additionally expected that a system trained using this methodology will only work correctly on images which are taken from similar sensors and altitudes.

The trained network is used with a sliding window based approach in order to detect multiple changes in a single image. A heuristic based approach is used for post processing in order to reject false positives and merge similar predictions.

4.1.1 Network Architecture

Since change is defined as maximum Euclidean distance between input samples, it is natural for the Network to work on a difference image directly. This difference image is

generated by taking the pixel-wise Euclidean distance between the clean and changed RGB images. This results in a greyscale image in which significant changes tend to appear as structured bright patches, whereas illumination differences tend to appear as smooth gradients. The task of the neural network is to help distinguish which of these areas contain actual changes, and which are insignificant.

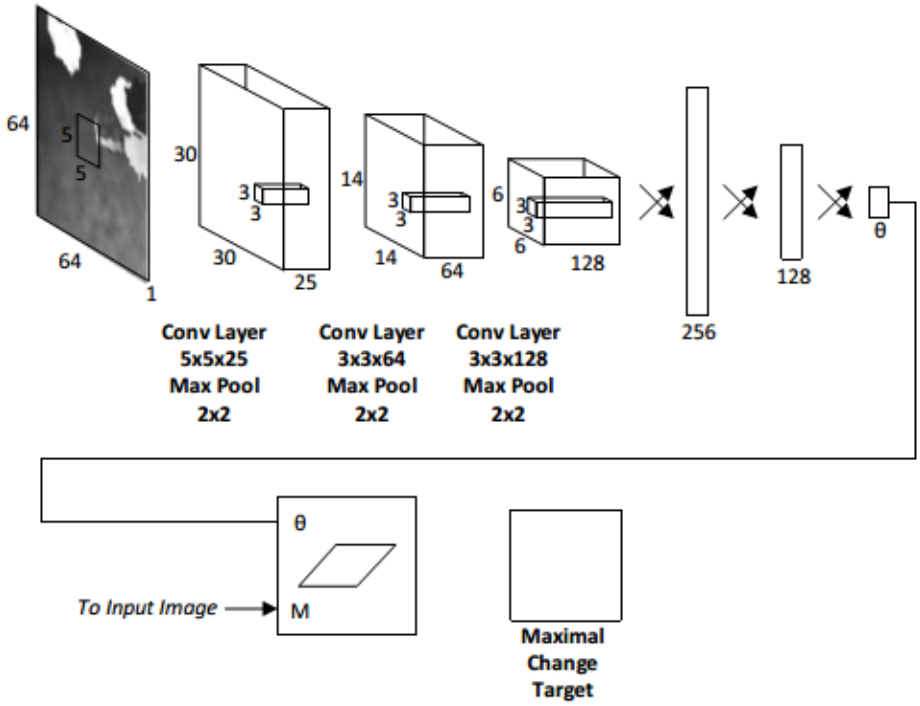


Figure 5: Semi-supervised Network Architecture.

The network architecture, displayed in Figure 5, consists of three convolution-pool layers followed by three dense layers and finally a spatial transformer. From first to last, the convolution layers consist of 25, 64, and 128 filters respectively. All convolution layers use 3x3 filters, and a leaky ReLU nonlinearity, with a leakiness factor, m, of 0.01. Each

is followed by a 2x2 max-pooling layer. The final pooling layer is followed by three consecutive dense layers consisting of 256, 128, and 2 units respectively. The first two dense layers utilize the same leaky ReLU nonlinearity, while the final dense layer, labeled θ, parameterizes the spatial transform and as such is left linear. Finally, the spatial transformer applies the θ transform to the input difference image, and outputs the result.

The transformation parameter θ, consists of two outputs corresponding to translation in the x and y directions. The scale parameter of the affine transformation is fixed at 50%, which approximately matches the size of the desired targets. For reasons conferred in the discussion, the fixed scale is necessary when using this methodology.

This particular network architecture was found through trial and error by starting with a very small network and growing it both in width and depth until the network began to overfit. At this point it was shrunk slightly to remove the overfitting. Both 5x5 and 3x3 filters were experimented with in the convolutional layers, as well as using a standard ReLU nonlinearity rather than the leaky ReLU. The final architecture represents the best of the tested architectures after training.

4.1.2 Network Training

Training data was generated from the Aerial Image Change Detection dataset described below. The 500 generated difference images were split into 80% training and 20% holdout sets, leaving 400 training images and 100 test images. Positive sample patches were extracted from both the training and holdout sets by randomly offsetting 64x64

windows from the center of the desired structure. For each of the images, 10 patches were extracted using random offsets, resulting in 4000 positive training samples and 1000 test samples.

Only positive samples are used during training in an attempt to get the network to overfit to the desired structures. Although the network doesn't receive explicit labels during training, the fact that only positive samples are used requires there to be an implicit label associated with each patch. This method is thus not fully unsupervised, and so the term semi-supervised is more accurate.

The loss function is defined as the mean square error between the predicted patch, and a maximally changed patch, which in the case of a scaled difference image is a fully saturated greyscale patch. Stochastic gradient descent with Nesterov momentum is used during the backpropagation pass to update the network weights.

4.1.3 Sliding Window and Post Processing

In order to detect multiple changes per image, a sliding window is employed. A 64x64 window is moved across the image, and a single location prediction is generated for each window position. Because the localization network is able to further narrow the location to a 32x32 box within a given window, it is unnecessary to visit every possible window location, and a strided approach was taken instead. A sequence of post-processing filters was applied to the predicted locations in order to reject false positives.

The first filter is based on the observation that the network tends to predict a translation of 0 if there are no structures present. As such, (5) shows that the first filter takes the set of all location predictions,P, and retains only the predictions which have a maximum translation of at least α. The new set of filtered predictions is labelled P_t. Maximum translation, $\boldsymbol{m}(P)$, is defined in (6) as the maximum magnitude of the x and y axis translations, where T^x and T^y are the translation in the x and y directions. It must be noted that using the α threshold is only useful when operating the sliding window where it is guaranteed that at least one window will contain the object off center. Otherwise, windows which happen to contain a centered object will be rejected.

$$P_t = \{P_{(i)} \mid \boldsymbol{m}(P_{(i)}) \geq \alpha \; \forall i\} \quad \textbf{(5)}$$

$$\boldsymbol{m}(P) = \boldsymbol{max}(|T^x|, |T^y|) \quad \textbf{(6)}$$

The next step is to filter out patches of flat texture which have low energy using (7). This filter takes the set of translation filtered predictions, P_t, as described above and retains only those with an energy above a threshold δ. The remaining predictions are placed in the set labeled P_e. The energy function, $\boldsymbol{e}(P)$, shown in (8) defines the energy of prediction P to be the mean squared error between p and μ, where p is the set of pixels in P, and μ is the average value of P. As with the first filter, the I subscripts denotes the i'th prediction in the given set.

$$P_e = \{P_{t(i)} \mid \boldsymbol{e}(P_{t(i)}) \geq \delta \; \forall i\} \quad \textbf{(7)}$$

$$\boldsymbol{e}(P) = \boldsymbol{mean}\left(\left(P_{(ij)} - \mu\right)^2\right) \quad \textbf{(8)}$$

The final filtering stage uses Algorithm 1 to combine predictions for which the overlap ratio is greater than β, and to remove any predictions for which fewer than γ original predictions agreed upon. The final set of predictions is taken to be the set of bounding boxes left over after post-processing.

4.2 Results

In the experimental setting, Python 3 was used as the programming language for all tasks. A combination of Nolearn [40], Lasagne [41], and Theano[42] API's were used to model the neural network. Theano is a tensor based computing API which allows for automatic differentiation, as well as GPU acceleration of tensor operations. Lasagne is a library which defines many common neural network layer types in terms of Theano tensors. Finally, Nolearn is a wrapper around Lasagne which makes it compatible with the scikit-learn API.

Algorithm 1: Merge and Overlap Filter

Data: P : Set of predicted bounding boxes
β : Minimum IoU for merging
γ : Minimum overlap score
Result: List of filtered predictions
$F \leftarrow [\ ]$;
$S \leftarrow [\ ]$;
for $p_1 \in P$ **do**
  $s_n \leftarrow 0$;
  $p_n \leftarrow P_1$;
  for $p_2 \in P - p_1$ **do**
    if $IoU(p_1, p_2) \geq \beta$ **then**
      $s_n \leftarrow s_n + 1$;
      $p_n \leftarrow avg(p_n, p_2)$;
      $P \leftarrow P - p_2$;
    end
  end
  $F \leftarrow F \| p_n$;
  $S \leftarrow F \| s_n$;
end
return $\{F_i \mid S_i \geq \gamma\}$

Both training and testing were performed using an Intel Core i5 6200U processor in addition to an Nvidia GeForce 940M GPU. The Neural Network Analysis and training were performed with a parallel batch size of 10 64x64 image patches. The Sliding window was performed sequentially using no GPU resources. Training the network took roughly 30 seconds per epoch, and took around 37 epochs to fully converge. This results in just under 30 minutes hours of training with the 4000 training patches. Running the analysis takes about 2 minutes 10 seconds to make predictions for the 1000 test patches.

4.2.1 Dataset

The Aerial Image Change Detection (AICD) dataset was used for the semi-supervised approach because it provides change images, as well as background images needed for

training [26]. This dataset consists of 100 simulated scenes each captured from 5 viewpoints giving a total of 500 samples. For every sample, the dataset provides the clean image, the image with a change, and the ground truth. Figure 6 shows an example for a single sample viewpoint. Although, each scene has both shadowed and un-shadowed variants, this experiment, only used the shadowed versions. The small number of images contained in this dataset makes it very important to keep the network complexity in check. With such a small dataset, too large of a network would not be able to fully train.

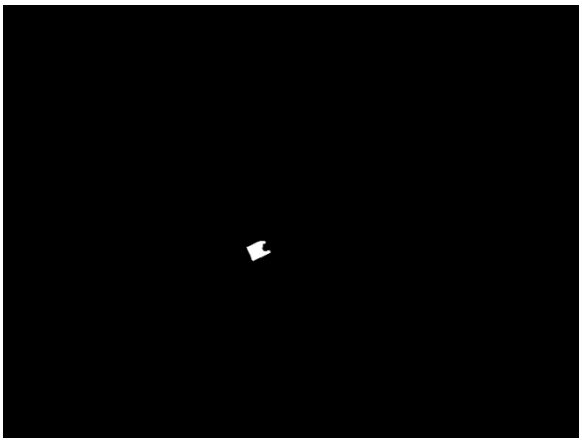
4.2.2 Localization Results

To characterize the localization ability of the network, all samples in the holdout set were passed through the network to collect predicted class and locations. Because no attempt is made to determine the relevancy of the predicted area, the prediction is irrelevant for negative samples. However if the sample is positive, then it is desired that the predicted area match the target area. Therefore, the Intersection Over Union (IOU) ratios between the predicted bounding box and ground truth bounding box were recorded for all positive samples. The IOU is a measurement which characterizes how well two areas overlap. As the name suggests, it is calculated by dividing the area of the bounding boxes geometric intersection by the area of their geometric union. In cases where multiple ground truth locations occur in a single patch only the maximum IOU was recorded.

(A) No Change

(B) Changed

(C) Difference Image



(D) Ground Truth

Figure 6: AICD Scene 94 - Viewpoint 0 With Changed Area Circled

As an example, Figure 7 shows a patch with multiple ground truth objects represented as circles; only the maximum IOU between the predicted square, and green circle is recorded. The localization performance was then calculated as the average IOU over all positive samples, along with the percentage of prediction areas which matched the ground truth.

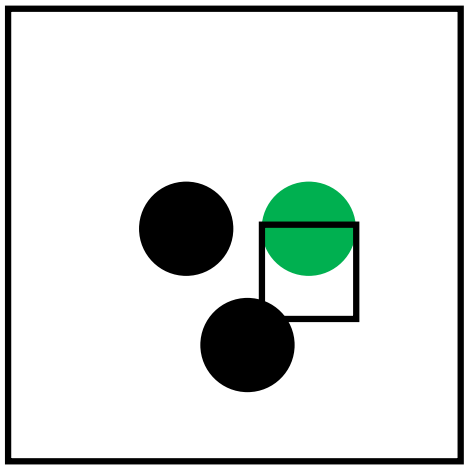


Figure 7: Localization Network IOU Example - The maximum IOU is recorded between the predicted square and green circle

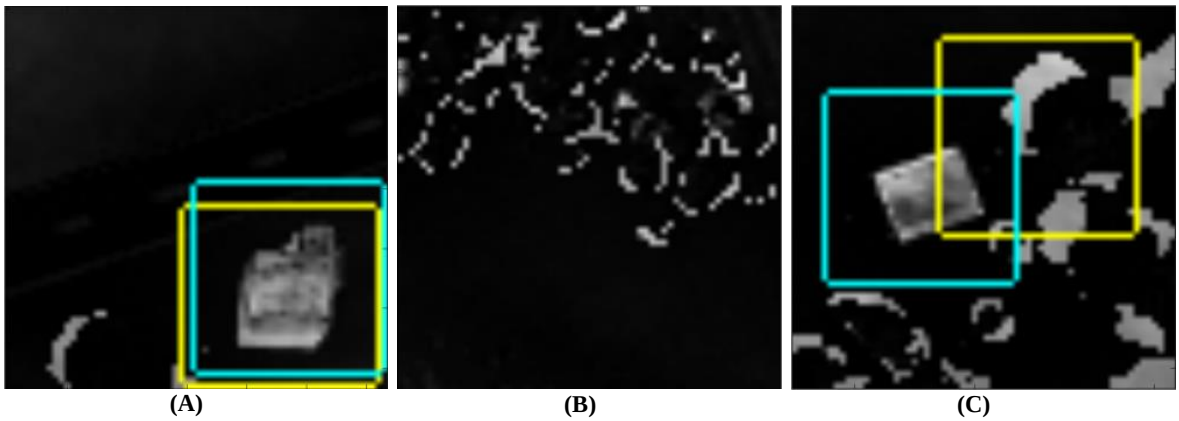
The average IOU of all predictions with their respective ground truths is shown in Table 1. The final column shows the percentage of predictions which match the ground truth location with an IOU greater than the given threshold. An IOU of 0.5 is the standard used in consumer image detection challenges such as the Pascal Visual Object Challenge [43], however in these challenges the target boxes tend to contain many thousands of pixels, and as such small deviations in bounding boxes don't effect the overall IOU significantly. In contrast, if a 32x32 px bounding box with an IOU of 0.5 is offset by only 4 pixels in both directions, then the IOU can decrease to 0.3. As such multiple IOU thresholds, including 0.1, 0.3, and 0.5, are tested.

The average IOU of 0.57 is lower than expected, but can be explained by the non-structural learning targets. Because the localization target is to simply find the "most changed" area, there is no way for it to determine which changed area it is desirable to find.

Table 1: Semi-supervised Localization Network Average IOU Results

Avg IOU	Match Threshold	Match %
0.57 ±0.2	0.1	99.4%
	0.3	87.0%
	0.5	62.1%

As an example, Figure 8 shows three qualitative examples of network predictions. In (A), the network has correctly identified the desired target changed area. Note however that there is very little background noise. In (B), the network has found an arbitrary location, but the energy threshold has determined that there is no object present. Finally, (C) shows that the network has trouble with background noise which is of similar size and shape to the desired structure. The network has performed the given task of finding the significantly changed area; however in this case, the given task doesn't match with what is actually desired. Cases similar to this can be expected to decrease the average IOU as well as the percentage of matched predictions.



(A) **(B)** **(C)**

Figure 8: Qualitative Unsupervised Network Predictions - Yellow is the predicted box, blue is the ground truth box

- **(A) The network has correctly localized the object of interest in a low noise environment**
- **(B) The network has correctly determined that no object of interest is present**
- **(C) The Network has identified an object in the wrong location due to clutter**

4.2.3 Energy Thresholding

In order to measure the ability of the energy function to reject false positives, a Receiver Operating Characteristics (ROC) curve was generated by varying the δ threshold. This curve plots the true positive rate against the false positive rate in order to visualize the tradeoff between finding all positive samples and finding no negative samples. Because only the predicted patch is seen by the energy function, the ground truth was taken to be whatever was present within the given predicted area, as shown in Figure 9. Note that although all patches contain a ground truth object, the test result is determined solely by what is contained inside the predicted area. For a predicted area to be considered as containing a ground truth object, it must overlap with the object with an IOU above a given threshold.

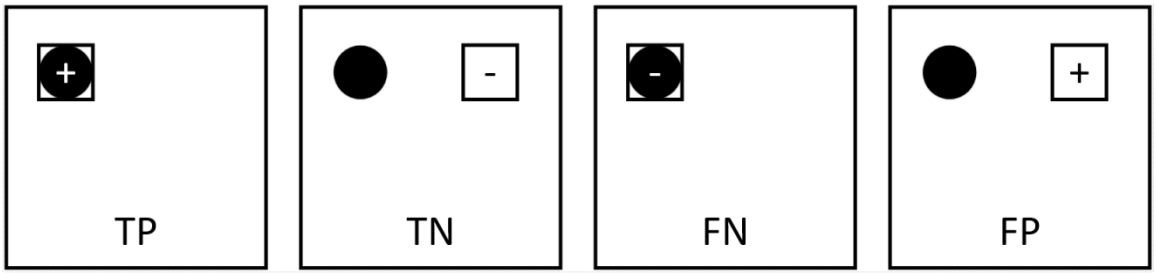


Figure 9: Prediction Truth Scenarios – The circle is the ground truth, the square is the predicted area, the sign is the class prediction, and the text at the bottom indicates the test result

The ROC curves shown in Figure 10 show that although the energy based relevancy measure isn't ideal, it performs significantly better than random given the locations provided by the network. When the network is used in the system, the translation of the transformation is also taken into account, which was expected to help even more.

In order to quantify the ROC performance as a scalar value, the Area Under the Curve (AUC) is calculated. Table 2 shows a summary of the calculated AUC scores for the tested ground truth IOU thresholds. With a minimum IOU of 0.5, the AUC of 0.788 shows that there is room for improvement, however given that the relevancy measure does not take the target's structure into account it is about as good as can be expected.

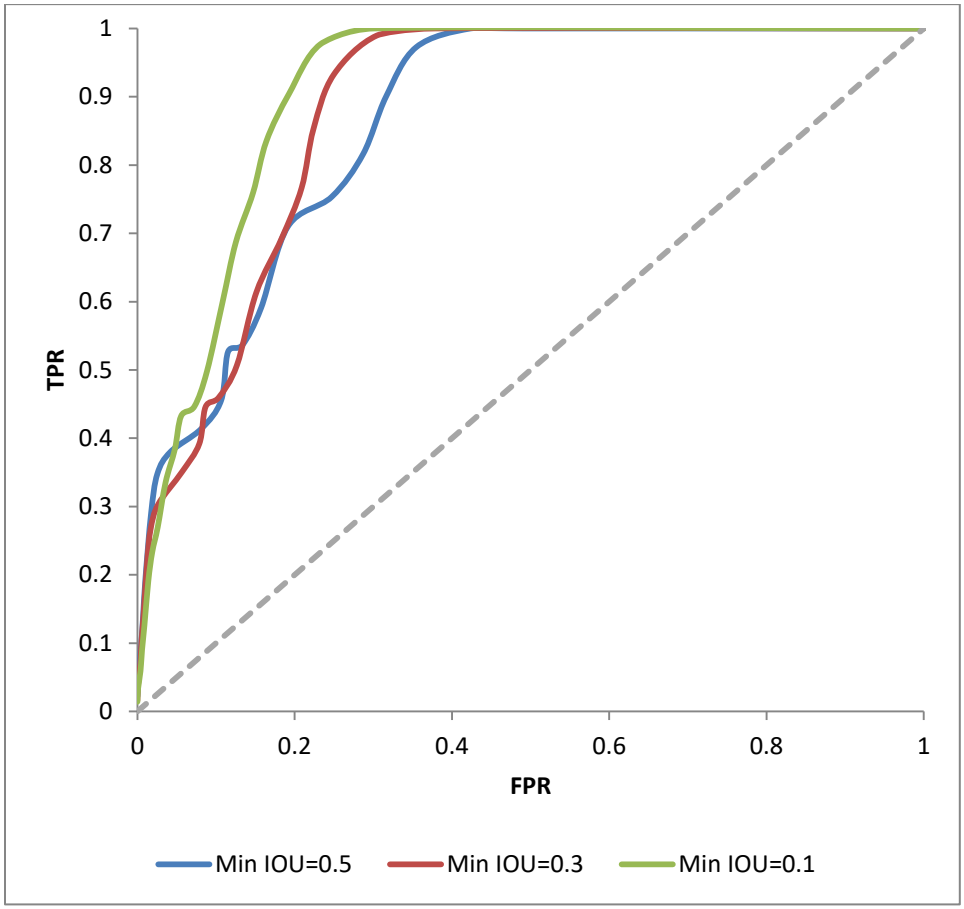


Figure 10: Unsupervised Energy Based Detection ROC

The Precision vs. Recall curve shown in Figure 11 was also generated for the different IOU thresholds. As can be seen, the semi-supervised approach loses precision steadily as

recall is increased. The mean Average Precision (mAP) for the network is listed alongside the AUC in Table 2. It can be seen that the minimum IOU threshold has a large impact on the precision of the network due to the fact that the average IOU of the location predictions is close to the threshold to begin with.

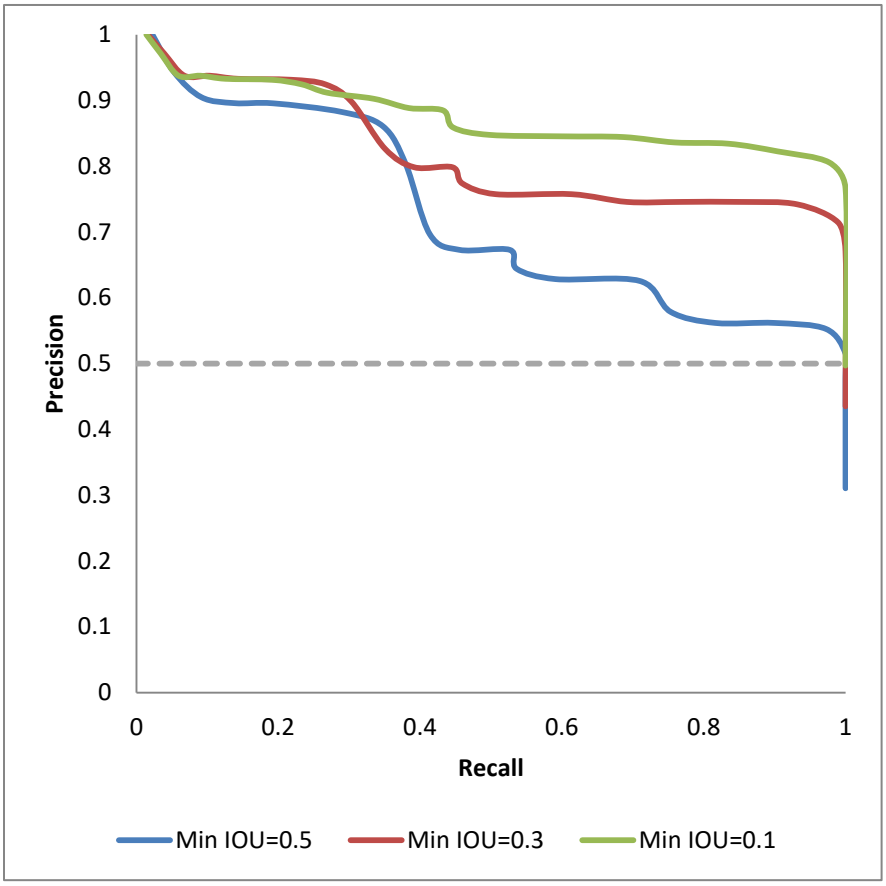


Figure 11: Semi-Supervised Energy-Based Detection Precision vs. Recall

Table 2: Unsupervised Area Under Curve and mean Average Precision

Min IOU	AUC	mAP
0.1	0.910	0.860
0.3	0.866	0.795
0.5	0.788	0.656

Table 3 gives the detailed truth counts for the semi-supervised network when the localization IOU threshold is set at 0.5. Metrics given include False Positive Rate (FPR), True Positive Rate (TPR), Precision (Pr), Specificity (Sp), F-Measure, and Percent Wrong Classification (PWC). See Appendix I for metric definitions.

Table 3: Detailed Energy Based Detection Characteristics (min IOU=0.5)

δ	TP	TN	FP	FN	FPR	TPR	Pr	Sp	F-meas	PWC
0.100	14	1379	0	607	0.00	0.02	1.00	1.00	0.04	30.4%
0.095	33	1377	2	588	0.00	0.05	0.94	1.00	0.10	29.5%
0.090	57	1373	6	564	0.00	0.09	0.90	1.00	0.17	28.5%
0.085	86	1369	10	535	0.01	0.14	0.90	0.99	0.24	27.3%
0.080	120	1365	14	501	0.01	0.19	0.90	0.99	0.32	25.8%
0.075	180	1355	24	441	0.02	0.29	0.88	0.98	0.44	23.3%
0.070	216	1344	35	405	0.03	0.35	0.86	0.97	0.50	22.0%
0.065	236	1321	58	385	0.04	0.38	0.80	0.96	0.52	22.2%
0.060	257	1267	112	364	0.08	0.41	0.70	0.92	0.52	23.8%
0.055	283	1232	147	338	0.11	0.46	0.67	0.89	0.54	24.3%
0.050	327	1220	159	294	0.12	0.53	0.67	0.88	0.59	22.7%
0.045	333	1195	184	288	0.13	0.54	0.64	0.87	0.59	23.6%
0.040	367	1162	217	254	0.16	0.59	0.63	0.84	0.61	23.6%
0.035	441	1115	264	180	0.19	0.71	0.63	0.81	0.67	22.2%
0.030	468	1038	341	153	0.25	0.75	0.58	0.75	0.65	24.7%
0.025	507	983	396	114	0.29	0.82	0.56	0.71	0.67	25.5%
0.020	559	943	436	62	0.32	0.90	0.56	0.68	0.69	24.9%
0.015	604	890	489	17	0.35	0.97	0.55	0.65	0.70	25.3%
0.010	621	795	584	0	0.42	1.00	0.52	0.58	0.68	29.2%
0.005	621	703	676	0	0.49	1.00	0.48	0.51	0.65	33.8%
0.000	621	0	1379	0	1.00	1.00	0.31	0.00	0.47	69.0%

4.2.4 System Results

When running the system as a whole there is an ambiguity as to what constitutes a true positive. Because an overlapping window is used, multiple positive predictions may map to the same ground truth location, and similarly a single prediction may overlap with

multiple ground truth locations. To handle this ambiguity, the Pascal Visual Object Challenge (VOC) method of only accepting a single prediction per ground truth is used. If multiple predictions match with a single ground truth, only the first is considered a true positive, while the rest are considered false positives. In this case a match is defined as two boxes with an IOU ratio greater than 0.5. More specifically, the truth counts are defined as follows:

TP – Number of ground truths which match at least one prediction

FN – Number of ground truths which don't match any predictions

FP – Number of predictions, which don't match an available ground truth

TN – Total Number of Predictions – (TP+FP+FN)

To measure the system wide performance the post processing sequence was run using various combinations of α, β, γ, and δ in order to generate Miss Rate vs False Positives Per Image (FPPI) curve. The patch agreement threshold γ was test with values of 1 and 2. It was found that at $\gamma = 1$, the system benefitted from a higher energy threshold 0f 0.02, while at $\gamma = 2$, a lower energy threshold of 0.01 worked slightly better.

Figure 12 shows the Miss Rate curve for four sets of parameters. The systems using two overlapping windows run slightly better than the systems with four overlaps due to the smaller possible number of negative samples which must be rejected. Even still, the high FPPI shows that the localization network selects irrelevant changes too often, and the energy and translation heuristics are not a strong enough method to reject the large number of negative samples created by the sliding window.

Figure 13 shows example results for four scenes using parameters $\alpha = 0.4, \beta = 0.5, \gamma = 2$, and $\delta = 0.01$. A common aspect of all four images is the number of false positives which pass the energy and translation heuristics.

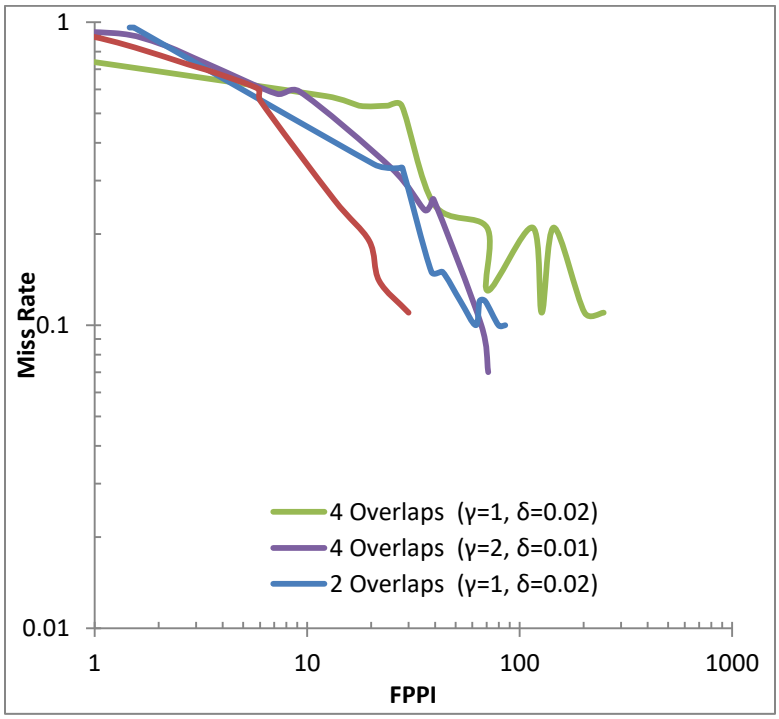


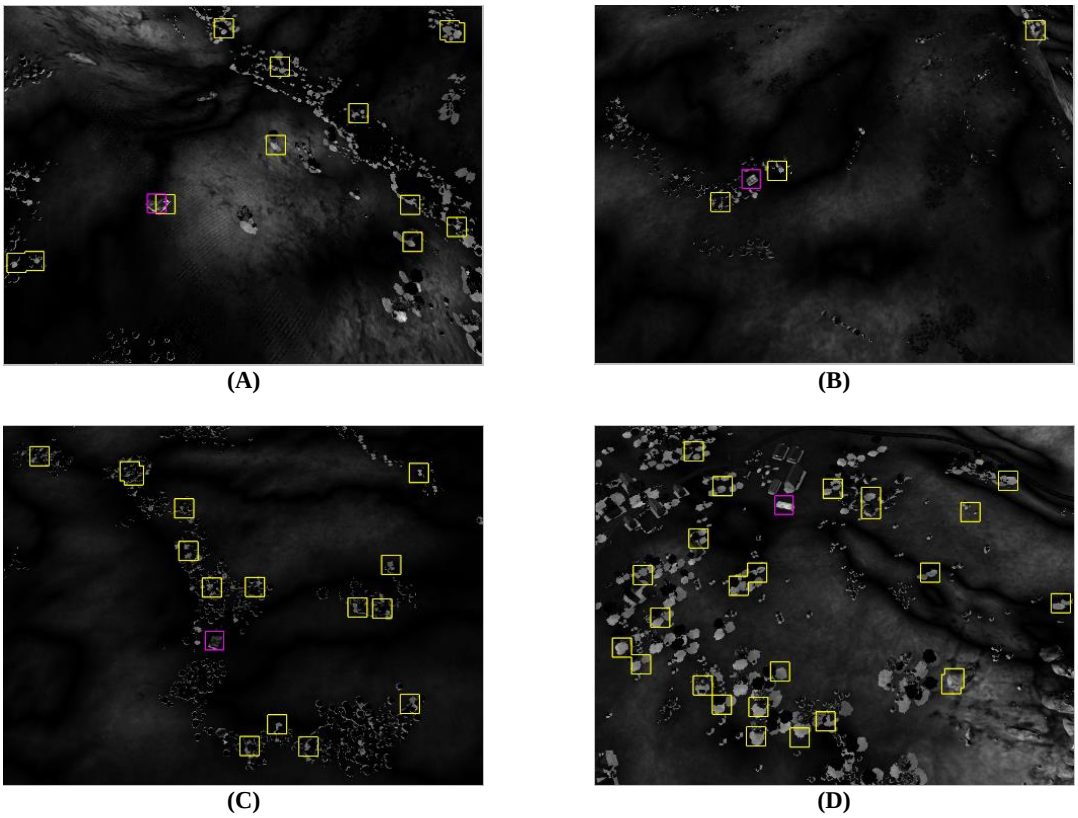
Figure 12: Semi-Supervised System Miss Rate vs FPPI

4.3 Discussion

4.3.1 Fixed Scale Parameter

When running the semi-supervised network, it became apparent that if the affine transformation scale parameter was left as a trainable parameter, then the network would simply learn to zoom onto a single bright pixel in the input patch. Given the definition

that change is represented by relative brightness in a difference image, this was the most convenient way for the network to find the maximal area of change, however it is obviously not the desired operation.



(A) **(B)**

(C) **(D)**

Figure 13: Semi-Supervised System Example Predictions
(A) The target has been found, in addition to false positives
(B) The target was missed, with few false positives
(C-D) The target is missed with many false positives

One potential method to counteract this without imposing a fixed scale would be to apply a penalty to predictions with small scales, however since the general size of the targets was known, it was easier to tune the scale parameter directly than to tune a penalty parameter.

4.3.2 Dataset Complexity and Overfitting

The AICD is very small dataset in terms of deep learning applications. The fact that the simulated images lack much complexity that would be present in a real image, along with the fact that there are only 500 scenes, means that anything but the simplest of networks would over fit.

Even with positive samples only, the network didn't quite operate as desired. Rather than overfitting to positive samples, it seems to find the largest bright area with little to no indication that it prefers the desired target over other differences. The simulated nature of the dataset may play a role in this outcome, as the shadows tend to have very sharp edges and lack subtleties found in real images. However this cannot be tested for using only the AICD dataset.

4.3.3 Semi-Supervised vs Unsupervised

As noted, although this methodology doesn't give the network explicitly labelled data, the fact that it only trains with positive samples implies that there must be an implicit label for each sample. Due to the fact that the network is trained with only positive samples, by definition it isn't unsupervised, however because no labels are used during training it isn't fully supervised either. Since the network doesn't completely lie in either unsupervised or supervised camps, the term semi-supervised was used.

4.3.4 System Window Overlap

When qualitatively assessing the semi-supervised network, it became apparent that it was able to parameterize transformations with larger translation components than the supervised counterpart. Because it proposes areas closer to the edges of the input patch, the overlapping window didn't need to be as dense as with the supervised network. This can be seen by the fact that the best performing version of the system uses only two overlaps vs the four used by the supervised system.

4.3.5 Real World Viability

Although the semi-supervised method was an interesting experiment it needs additional training on large scale datasets before it can mature into a viable change detection solution. The network is able to locate changes within an image, but without any reference to what constitutes a relevant change, the network will not be able to determine this distinction on its own.

This can be seen by the large number of predicted locations which correspond to an irrelevant change. Since relevant changes tend to be greatly outnumbered by irrelevant changes, such as illumination differences, a method of determining a change's relevancy is needed. Given that fact that implicit labels are needed anyways, it may be more productive to train a supervised system.

Chapter 5. Fully Supervised Approach

In this section a fully supervised method is presented which builds on the semi-supervised approach described in the previous chapter. Rather than trying to find an arbitrary area of maximum change, this methodology is paired with a binary classifier in order to detect specific targets. This not only increases localization performance, but also removes the need for the heuristics and replaces them with an explicit detection network.

Rather than finding changes directly, this methodology instead detects specific targets. In order to detect changes, a set of registered images is required. If a target is detected at the same position in both images, then no change has occurred, however if the target is only present in a single image, then a change has occurred.

5.1 Methodology

The supervised approach aims to combine the tasks of localization and detection by using a Spatial Transformer Network. Rather than attempting to find an arbitrary area of maximal change, this network is trained to find specific targets in a grayscale aerial image. A neural network binary classifier is appended to the transformation, and used as a detector. The entire network is trained end to end using multi task regression.

The trained network is then integrated with a sliding window similar to that used with the semi-supervised approach. Rather than heuristically rejecting false positives, the trained detector is used instead.

As with the semi-supervised methodology, it is expected that a system which uses this methodology will only work on images which contain similar targets taken from comparable altitudes using the same type of sensors. If these limitations are not conformed to, then the target signatures will be too dissimilar for the detection network to reliably find. In order to make the methodology generalizable, a dataset containing the desired target types and altitudes must be used for training.

5.1.1 Network Architecture

The network shown in Figure 14 is based on the network used in the semi-supervised method with three exceptions. First, the scale parameter of the affine transformation does not need to be fixed at a predefined value. Second, the localization target is now a specific structure rather than an arbitrary change. And third, a second spatial transformer and detection network have been added.

The first spatial transformer branch performs the transformation directly to the 64x64 input patch and outputs a zoomed in image patch labeled “localization output”. This branch is trained to reconstruct the 32x32 patch centered on the structure of interest as described in the following section.

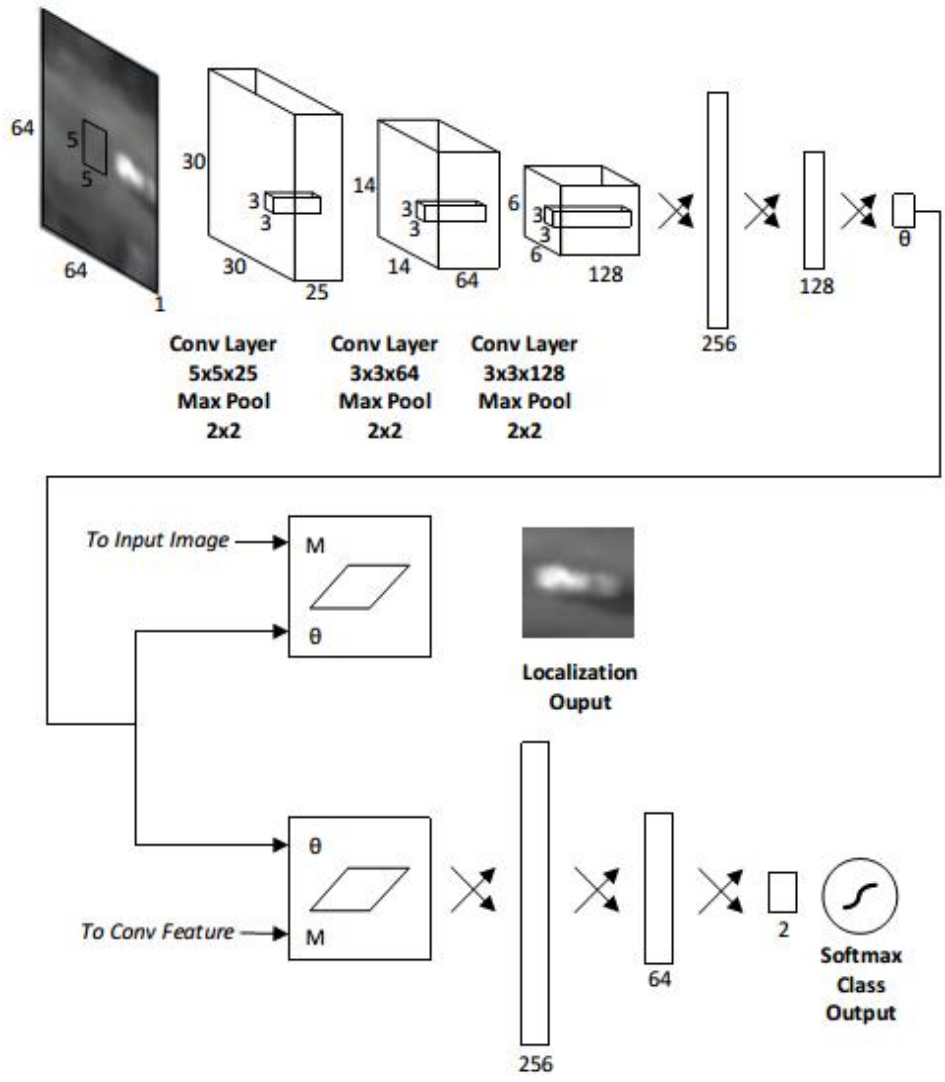


Figure 14: Supervised Network Architecture

The second branch performs an identical transformation onto the feature map output by the final convolution layer. This results in an abstract "zoomed in" feature, which is then fed to a binary softmax classifier for use in detecting relevant changes. This branch is trained to minimize the categorical cross-entropy between the predicted relevancy and the

target relevancy. As with the localization portion of the network, all dense layers in the detector use leaky ReLU nonlinearities with a leakiness of 0.01.

Two versions of this network were considered. The first version leaves the scale parameter of θ as a learnable unit, and connects the feature map input of the detector spatial transformer to the final convolution map. In contrast, the second version fixes the scale parameter at 0.5, and connects the detector transformer to the output of the first convolution layer. As will be discussed in the results, this fixed-scale network performed better in multiple ways.

5.1.2 Training

The total loss function used for training is the weighted sum of the two task losses as shown in Equation (9), where M is the pixel-wise mean squared error function, C is the categorical crossentropy function, T_{det} is the target detection relevancy in {1,0}, T_{loc} is the target location patch, and P_{det} and P_{loc} are the predicted relevancy and location patch respectively. In order to ensure that neither the detection nor localization task overpowered the total loss, the objective function weighted each task loss with hyper-parameters λ_{det} and λ_{loc}.

$$L = \lambda_{det} \cdot C(T_{det}, P_{det}) + \lambda_{loc} \cdot T_{det} \cdot M(T_{loc}, P_{loc}) \quad (9)$$

Note that the localization loss term is multiplied by the target detection relevancy. This ensures that the localization loss term only contributes to the total loss for samples which actually contain a relevant target ($T_{det} = 1$). Conversely, the localization term is

irrelevant for samples which don't contain a target ($T_{det} = 0$), and as such must be suppressed.

To extend this strategy from small 64x64 patches where the STN operates to large images, a sliding window approach is used. First the image is split into overlapping 64x64 pixel patches. The network is then used to generate a single localization and relevancy prediction per patch. Post-processing is used to filter out bad predictions.

5.1.3 Post processing

The post processing for the fully supervised system is based on a similar principle to that of the semi-supervised system. The difference is that rather than heuristically rejecting false detections using energy and minimum translation, the class score output by the second network branch is used.

First, (10) is used to remove all predictions with a detection relevancy score lower than α. This filter retains a set of predictions P_c, consisting of all predictions, P, which have a class score above a threshold α. The $P_{(i)}$ and $C_{(i)}$ represents the i'th prediction and its class score respectively.

$$P_c = \{P_{(i)} \mid C_{(i)} \geq \alpha\} \tag{10}$$

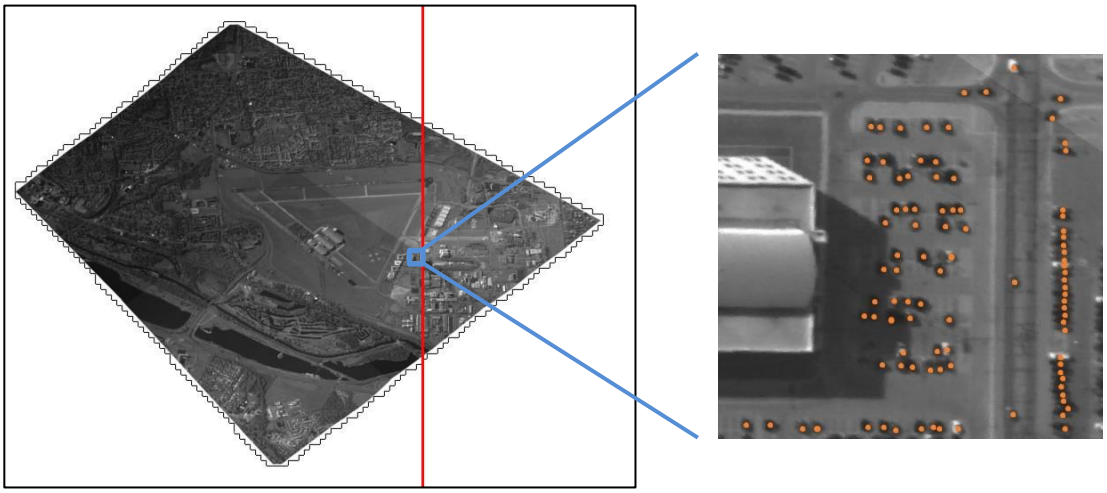
Then, the same merge and agreement filter is used as the semi-supervised system. Algorithm 1 is used to merge predictions for which the overlap ratio is greater than β, and remove any remaining predictions for which fewer than γ original predictions agreed upon. The final set of predictions is taken to be the set of bounding boxes left over after post-processing.

5.2 Supervised Results

5.2.1 WPAFB Dataset

The Wright Patterson Air Force Base (WPAFB) Dataset was used for the supervised experiment because it is one of the few publicly available aerial image databases [44]. This dataset consists of over 1600 greyscale aerial images taken above the WPAFB over the course of 20 minutes. Each image is roughly 30,000 by 20,000 pixels covering about $64\ \mathrm{Km}^2$. Ground truth locations are provided for moving vehicles only. Because nonmoving vehicles are not labelled, this dataset is not well-suited for detection evaluation.

To obtain training and test data for our experiments, a single scene from the dataset was manually labelled to define the location of every vehicle in the frame. Windows of 512x512 pixels were displayed on screen at a resolution of approximately 100 pixels per inch, and ground truth points were generated from mouse clicks. Figure 15 shows the full scene as well as a labeled sub window. In some cases, such as parking lots, where cars were close enough that it was difficult to determine where one ended and another began, a best guess was taken. Although it is possible that some points are mislabeled, it is expected that the vast majority are accurate. This process resulted in roughly 4,500 labeled vehicles which were used for training.



(A) (B)

Figure 15: Full WPAFB Dataset Scene
(A) Full Scene Overview - The vertical line corresponds to the initial train-test split at column 19,968
(B) A 512x512 image chunk labeled with ground truth center points

For each target sample in the training set a 64x64 patch was randomly offset from center by up to 16px in both cardinal directions. An equal number of 64x64 patches which contained no labeled locations were then selected for negative samples. For positive samples, the localization target was set to the 32x32 patch centered perfectly on the location, while for negative samples the localization loss does not contribute towards the objective. The detection target is set to 1 for positive samples and 0 for negative.

As with the semi-supervised network, both training and testing were performed using an Intel Core i5 6200U processor with an Nvidia GeForce 940M GPU. Training the network took roughly 90 seconds per epoch, and took around 100 epochs to fully converge. This results in just less than 3 hours to train on three of the four K-Fold splits. Generating predictions and running post processing for an entire WPAFB scene takes roughly an hour and a half. It is important to note that parallel processing of the

512x5123 image chunks could speed up the analysis. Running the analysis takes about 2 minutes 10 seconds to make predictions for the 1000 test patches.

5.2.2 Localization Results

For the initial localization test, column 19,968, shown as a vertical red line in Figure 15, was used to split the scene into roughly 80% training and 20% testing samples. Figure 16 shows a set of typical location predictions for the free scale network. The three columns show the input patch, the target patch, and finally the predicted patch received from the network. The yellow box in the first column displays the bounding box for the location prediction. The class score, as well as the affine prediction are shown to the right of each prediction.

The first row (A) shows the network predicting a true positive where both the class and the location are correct. In the second row (B), the network has detected the target correctly, however rather than translating a small box, it has kept the box mostly centered and simply grown it until the car is in bounds. The third prediction (C) shows the network correctly predicting that no cars are present, even though there are structures in the bottom left corner. In this case, the location output can be completely ignored. Finally, the last row (D) shows the network predicting a false positive. In this case, it is easy to see that the network was confused by the bright patch on the left side.

The main issue with the free-scale network is that rather than translating a small box to the target, it instead simply grows the box until the target is within bounds. This can be

seen in Figure 16 (B) and (C) where the bounding box is significantly larger than the target requires.

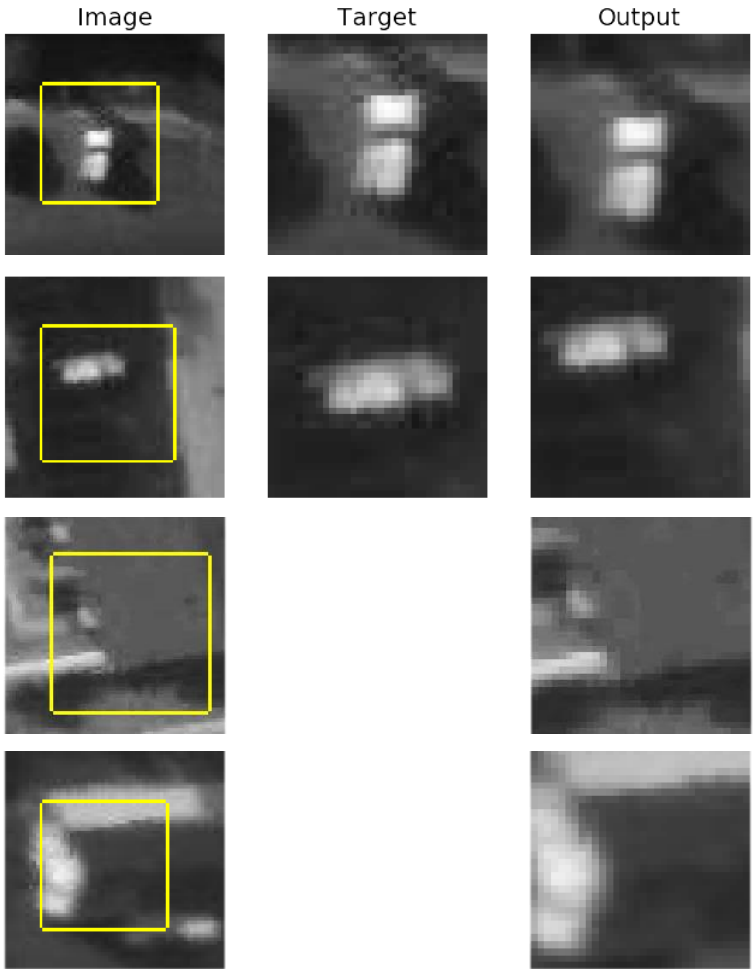


Figure 16: Localization and Detection Examples for Free-Scale Network
Sub figures show that the network has :
(A) detected and localized the car well.
(B) detected the car correctly, but the localization is unsatisfactory
(C) correctly determined that nothing is present
(D) incorrectly identified a car where none is present

The inability for the network to correctly learn the scale parameter was thought to be due to two factors. First, there might simply not have been enough training data, and second, the detection task may have been overpowering the localization task by growing the bounding box to get a larger context. To reduce these issues, the scale parameter was

fixed at 0.5 as determined by the general size of all vehicles within the dataset. This eliminates the competition between the networks, however it still leaves the issue of enough context making its way to the detection layers. The second issue was alleviated by connecting the detection transformer to the first convolution layer rather than the last, thus the detector receives a full 30x30 feature map rather than the smaller 6x6 feature.

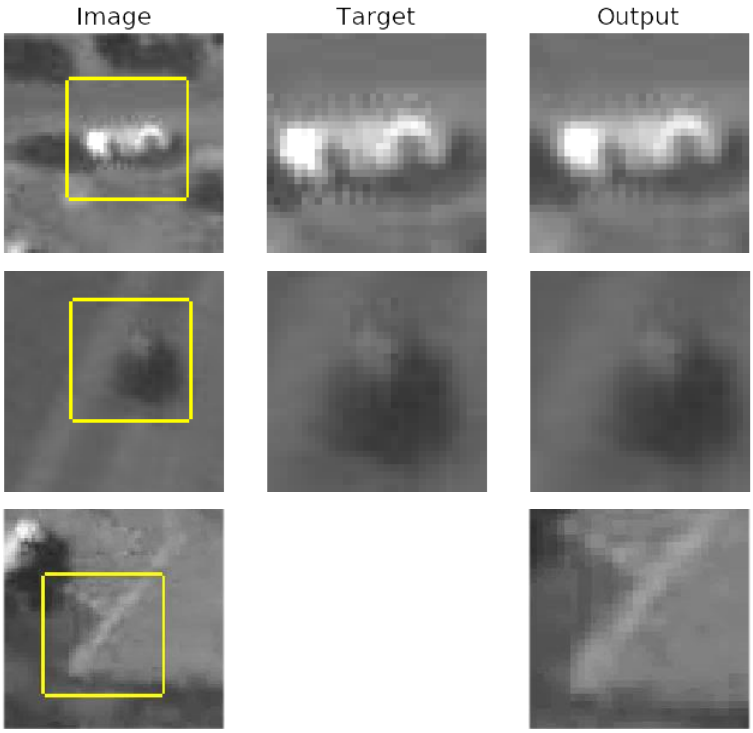


Figure 17: Localization and Detection Examples for Fixed-Scale Network
Sub figures show that the network has :
(A) detected and localized a centered car correctly
(B) detected and localized an offset car correctly
(C) found an interesting patch and determined that nothing is present

Localization metrics were calculated similarly to the semi-supervised system. The average IOU, as well as the percentage of matched locations were both considered. Table 4 and Table 5 show the initial test localization results for the fixed-scale and free-scale networks respectively. The "Total" column shows the average IOU of all predictions

with the ground truth. The percentage column shows the percentage of predictions which matched with a ground truth object. In all cases, matching with a ground truth object is defined as the 32x32 prediction overlapping with the 32x32 ground truth target with IOU greater than that specified by the given IOU threshold. As can be seen, the fixed scale network greatly outperforms the free-scale network in terms of localization. Not only are the average IOUs consistently greater, but also the percentage of matching predictions stays high even when the minimum IOU threshold is increased. Due to its better performance, only the fixed scale network results are discussed moving forward.

Table 4: Fixed Scale Network Average IOU Localization Results

Fixed Scale		
min IOU	Total	%
0.1	0.66 ± 0.2	100%
0.3		96.8%
0.5		82.7%

Table 5: Free-Scale Network Average IOU Localization Results

Free Scale		
min IOU	Total	%
0.1	0.46 ± 0.2	99.8%
0.3		80.8%
0.5		36.5%

To alleviate concerns of dataset bias, K-fold cross validation was performed with K=4 to ensure that the network performed similarly across the entire scene. Folds were generated using random sampling of 512x512 labelled chunks, and patch samples were extracted from their respective folds.

Table 6 shows the K-fold localization results, where the parenthesized values indicate the match percentage for a set of random predictions with a given IOU threshold.

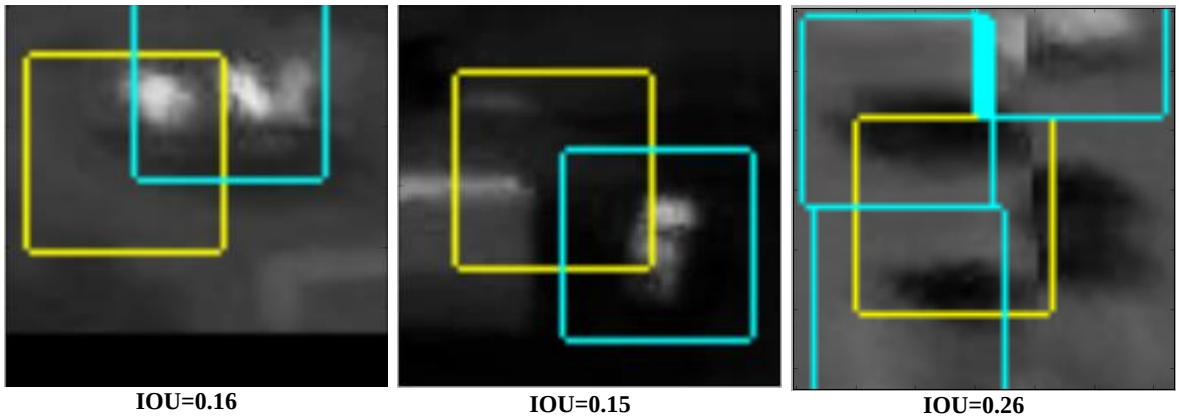
These results illustrate that, on average, the predicted locations overlap with the ground truth with an IOU greater than 0.5 for almost 78% of predictions. If the ground truth overlap ratio is decreased to 0.3, then nearly 94% of predictions match. Using an IOU threshold of 0.1 doesn't give much insight into the reliability of predictions due to the fact that even random predictions achieve a 90% match rate.

Table 6: Fixed Scale Network K-Fold Localization Results

		Percent Matching		
Fold	**Avg IOU**	**min IOU 0.1**	**min IOU 0.3**	**min IOU 0.5**
1	0.65 ± 0.19	99.0% (89%)	93.9% (52%)	81.4% (28%)
2	0.64 ± 0.19	99.0% (89%)	94.3% (59%)	77.6% (27%)
3	0.62 ± 0.19	99.4% (91%)	93.9% (57%)	74.5% (27%)
4	0.63 ± 0.19	98.9% (90%)	93.3% (60%)	77.5% (27%)
Total	0.63 ± 0.19	99.1% (90%)	93.8% (57%)	77.8% (27%)

Although the average IOU of 0.63 is acceptable, it is still interesting to analyze some of the ways that the localization can be wrong. As such, Figure 18 shows a set of typical samples from the fixed-scale network which were detected correctly but had a low IOU score with the ground truth. As can be seen in (A), a significant cause for a missed prediction area seems to be that the predicted bounding box encapsulates the object closer to its edge than its center. The ground truth on the other hand is expecting the bounding box to be centered. The second sample (B) shows that in some cases the location prediction seems to be confused by background clutter, and coincidentally the detector predicts that a target is present. Finally, the last case (C) shows multiple objects within the input patch: the labelled vehicle as well as the vehicle on the right border. The

network localizes somewhere between them rather than fully committing to a single object.



IOU=0.16 (A) **IOU=0.15 (B)** **IOU=0.26 (C)**

Figure 18: Correct Detections with Low IOU - Yellow boxes are the predictions, blue boxes are the ground truth

5.2.3 Network Weights

When training a neural network it is common to examine the learned weights to gain insights into what the network has learned. It is widely known that a well-trained network should have first layer weights which look like Gabor filters. As a simple sanity check, the first layer weights for the shared convolutions, as well as the first dense layer in the localization and detection portions of the network were analyzed using the network trained on the first k-fold split.

Figure 19 shows the weights learned by the first convolutional layer. Gabor like features are discernable within these 5x5 filters, however there is also a fair amount of pixilation present.

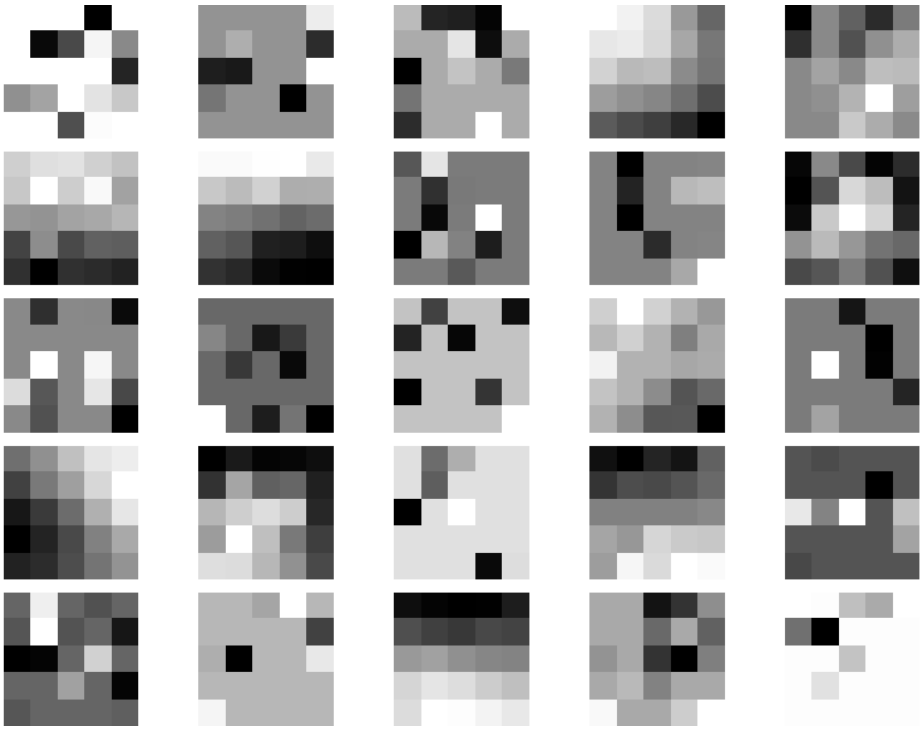


Figure 19: First Convolution Layer Weights

Similarly Figure 20 shows the weights learned by the first 144 units of the first dense layer in the localization network. These dense units each connect to 128 6x6 feature maps output by the final shared pool layer in the convolutional portion of the network. The visualization shows the mean weight values for each unit. These 6x6 weight maps show a set of abstract features, so it is hard to determine which are useful. A number of the units seem to have learned flat weights, where single pixel is the same value indicating dead weights or overfitting.

Figure 20: First Dense Localization Layer Weights

Finally Figure 21 shows the weights learned by the first 144 units of the dense detection network. Each of these units connects to a 25x30x30 feature map which is a spatially transformed version of the first convolutional layer. As can be seen, many of the units haven't improved substantially from the random initialization. This indicates that using fewer units in this layer would be beneficial.

Figure 19 - Figure 21 all lean towards the fact that the modelling capacity of the network may be too large for the given dataset and task. Using fewer filters in the convolution

layers and fewer dense units in the dense layers would allow the network to learn a better model of the data with fewer noisy weights.

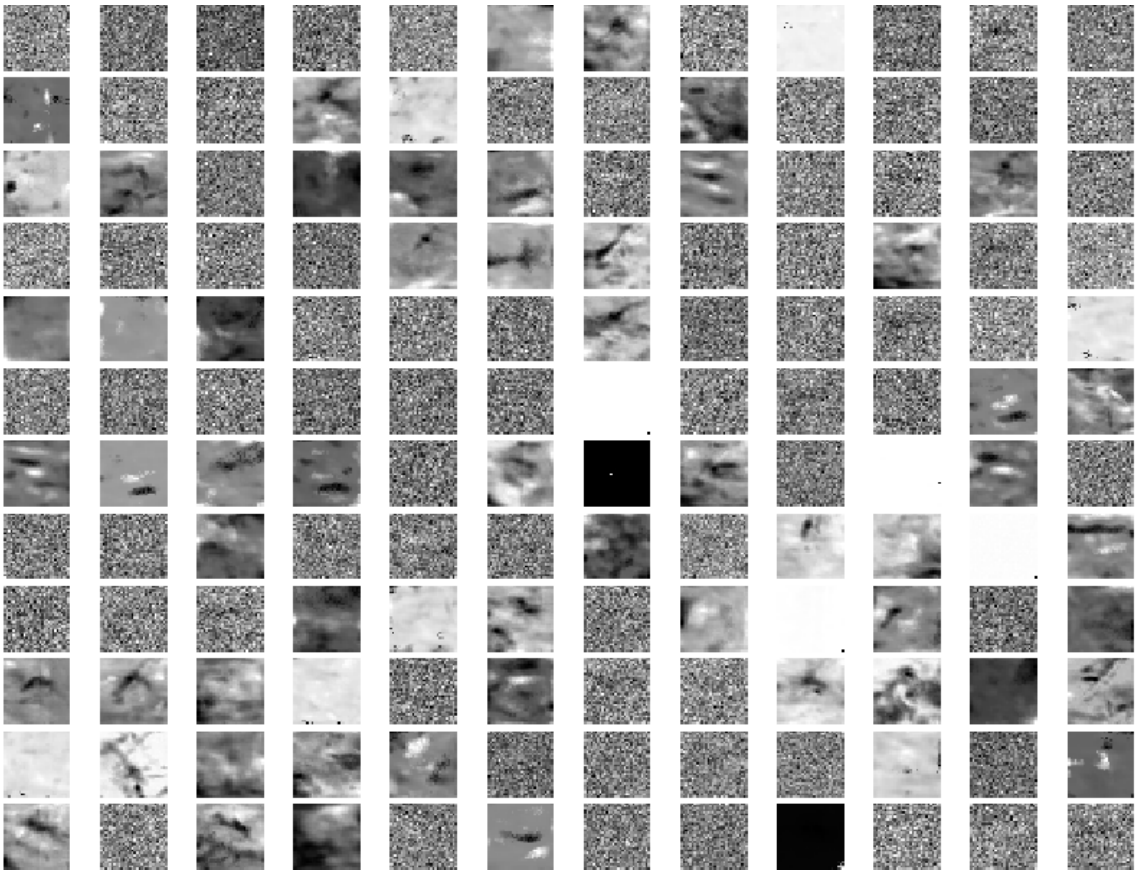


Figure 21: First Dense Layer Detector Weights

The noise in the weights can also indicate that the network has overfit, however the fact that the validation loss closely follows the training loss as seen in Figure 22, along with the fact that the network performs similarly across all dataset splits show that overfitting is not an issue. Regardless, shrinking the network, applying more regularization, and using a larger dataset would all remove this concern.

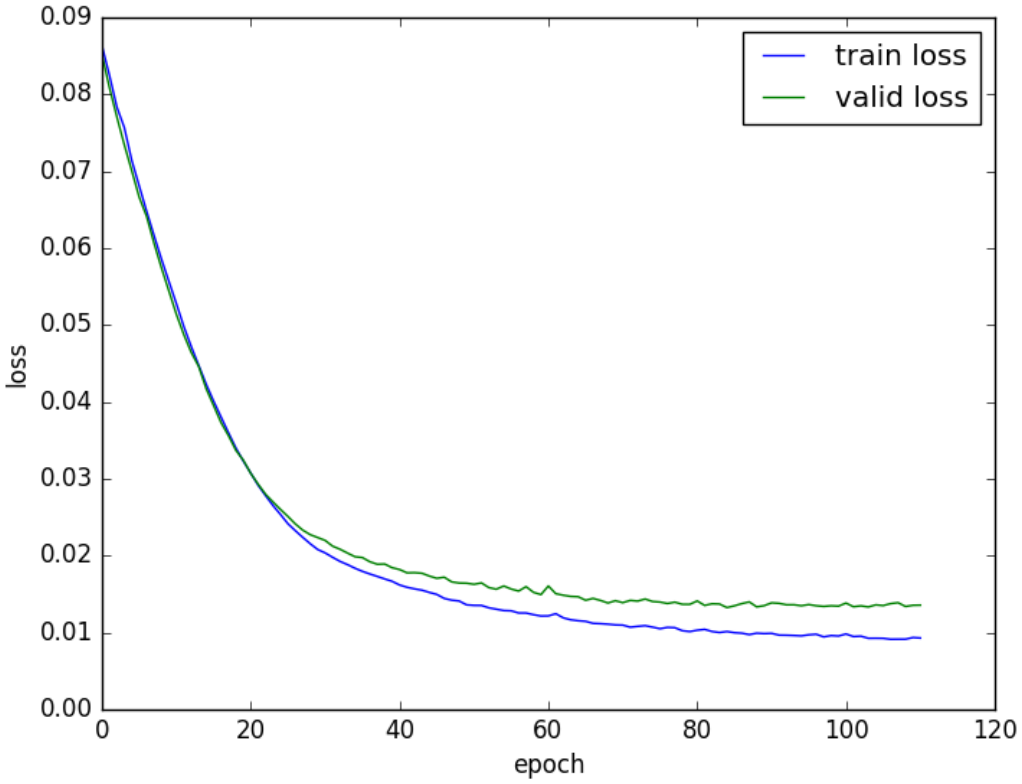


Figure 22: Training Loss History

5.2.4 Detection Results

Moving on to detection results, Figure 23 shows the ROC curves generated for the four data splits given a minimum IOU threshold of 0.5. As was expected, the network performs similarly for all four splits. Table 7 shows the AUC results for each of the splits, as well as each of the minimum IOU thresholds. The average AUC score of 0.91 shows that the detector performs well when the localization net correctly predicts the area.

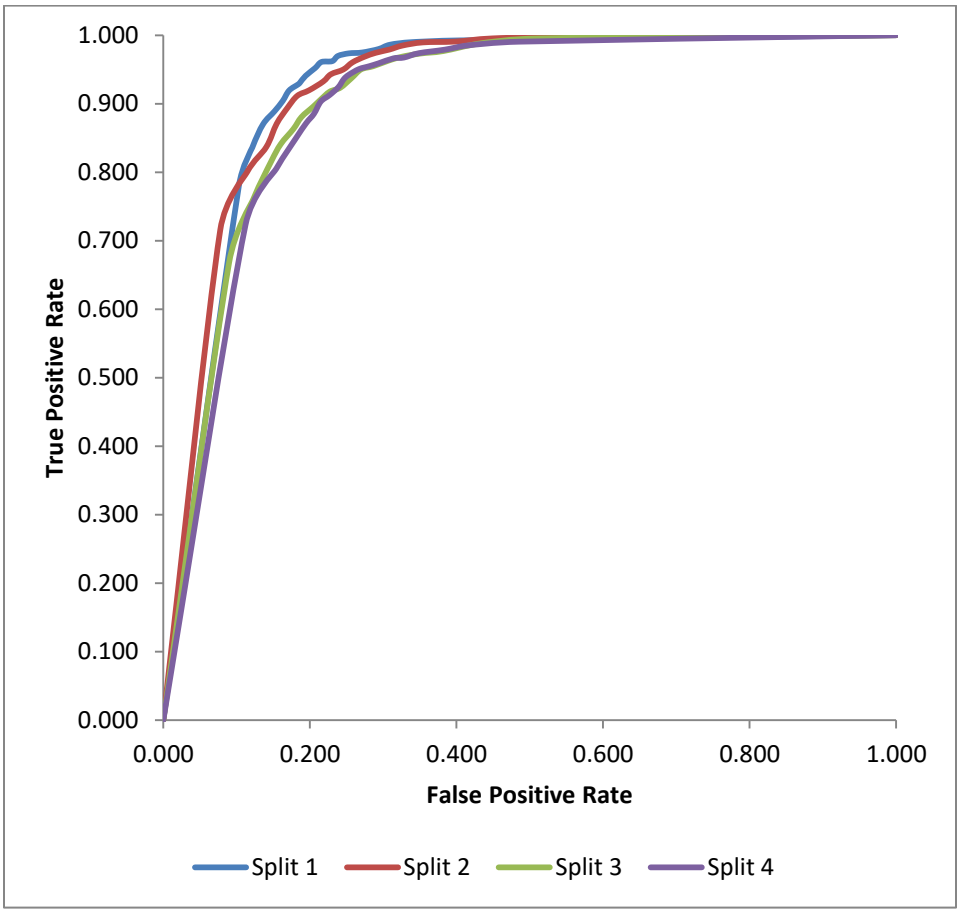


Figure 23: Supervised Detector Receiver Operating Characteristics

Table 7: Supervised Detector ROC Areas

AUC Scores

	Min IOU		
Split	**0.1**	**0.3**	**0.5**
1	0.936	0.926	0.919
2	0.935	0.934	0.923
3	0.928	0.929	0.906
4	0.926	0.917	0.898
Avg	0.931 ± .005	0.927 ± .007	0.912 ± .011

Detailed results for the fixed scale network are shown in Table 8. The truth counts in this table reflect the predictions of the detection output as it corresponds to the predicted area.

The same metrics used with the semi-supervised network are given. Refer to Appendix I for specific definitions.

Finally, to get a sense of how the localization network and detection network interact, the precision vs. recall curves for the detector were generated. Figure 24 shows these curves for all four splits with a minimum IOU set at 0.5, while Table 9 shows the mAP scores for all splits and all minimum IOU thresholds. Note that the reason Split 1 reaches a precision of 1.0 is due to the single true positive detection when α equals 1.

Table 8: Detailed Network Test Results For Split 1 (min IOU=0.5)

α	TP	TN	FP	FN	FPR	TPR	Pr	Sp	F-Meas	PWC
1	1	1154	0	791	0.00	0.00	1.00	1.00	0.00	40.6%
0.95	618	1035	119	174	0.10	0.78	0.84	0.90	0.81	15.1%
0.9	**662**	**1014**	**140**	**130**	**0.12**	**0.84**	**0.83**	**0.88**	**0.83**	**13.9%**
0.85	**689**	**997**	**157**	**103**	**0.14**	**0.87**	**0.81**	**0.86**	**0.84**	**13.4%**
0.8	**704**	**979**	**175**	**88**	**0.15**	**0.89**	**0.80**	**0.85**	**0.84**	**13.5%**
0.75	**716**	**966**	**188**	**76**	**0.16**	**0.90**	**0.79**	**0.84**	**0.84**	**13.6%**
0.7	**728**	**956**	**198**	**64**	**0.17**	**0.92**	**0.79**	**0.83**	**0.85**	**13.5%**
0.65	**735**	**942**	**212**	**57**	**0.18**	**0.93**	**0.78**	**0.82**	**0.85**	**13.8%**
0.6	**737**	**939**	**215**	**55**	**0.19**	**0.93**	**0.77**	**0.81**	**0.85**	**13.9%**
0.55	**745**	**930**	**224**	**47**	**0.19**	**0.94**	**0.77**	**0.81**	**0.85**	**13.9%**
0.5	754	915	239	38	0.21	0.95	0.76	0.79	0.84	14.2%
0.45	761	905	249	31	0.22	0.96	0.75	0.78	0.84	14.4%
0.4	762	888	266	30	0.23	0.96	0.74	0.77	0.84	15.2%
0.35	768	879	275	24	0.24	0.97	0.74	0.76	0.84	15.4%
0.3	771	862	292	21	0.25	0.97	0.73	0.75	0.83	16.1%
0.25	772	841	313	20	0.27	0.97	0.71	0.73	0.82	17.1%
0.2	776	815	339	16	0.29	0.98	0.70	0.71	0.81	18.2%
0.15	781	797	357	11	0.31	0.99	0.69	0.69	0.81	18.9%
0.1	784	762	392	8	0.34	0.99	0.67	0.66	0.80	20.6%
0.05	786	688	466	6	0.40	0.99	0.63	0.60	0.77	24.3%
0	792	0	1154	0	1.00	1.00	0.41	0.00	0.58	59.3%

Table 9: Supervised Network Mean Average Precision

	mAP Scores		
	Min IOU		
Split	**0.1**	**0.3**	**0.5**
1	0.936	0.918	0.888
2	0.914	0.902	0.830
3	0.917	0.899	0.790
4	0.907	0.876	0.782
Avg	0.918 ± .013	0.899 ± .017	0.823 ± .048

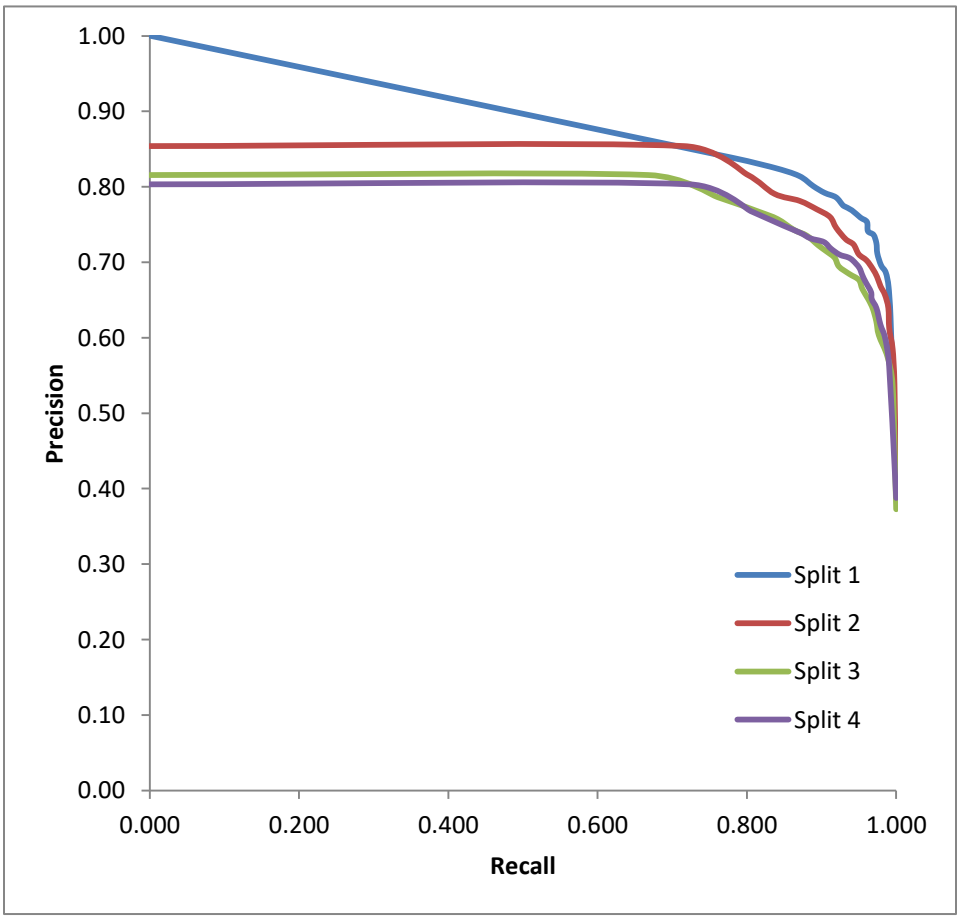


Figure 24: Supervised Precision vs Recall Curve

5.2.5 System Results

Due to the enormous size of the WPAFB scene, the sliding window system evaluation was performed in chunks of 512x512 pixels. Because the scene doesn't uniformly fill the rectangular image, there is a relatively large white border which can be seen in Figure 15. Rather than waste computations on these areas, all chunks which did not contain at least a single non-white pixel were completely ignored. As with the semi-supervised approach, the system performance was measured using the Miss Rate vs. False Positive Per Image (FPPI) curve.

The sliding window approach was performed using the network and holdout set for the first k-fold split. Varying the post processing parameters α, β and γ showed that the best Miss Rate performance was obtained using $\beta = 0.5$ and $\gamma = 1$. By varying the detection threshold α, the Miss Rate Curves shown in Figure 25 were generated.

This figure shows that the ideal overlap is two, increasing to four overlaps greatly increases the runtime, but doesn't significantly improve performance. That being said, the FPPI at which the miss rate begins to decrease indicates that the detector is not strong enough to deal with the huge number of negative samples that it must correctly reject. For the first k-fold split, running the sliding window with two overlaps generates nearly 70,000 windows, only 973 of which contain a positive target. Even if the detector was able to correctly reject 95% of negative windows, it would still be expected to produce around ten false positives per 512x512 section.

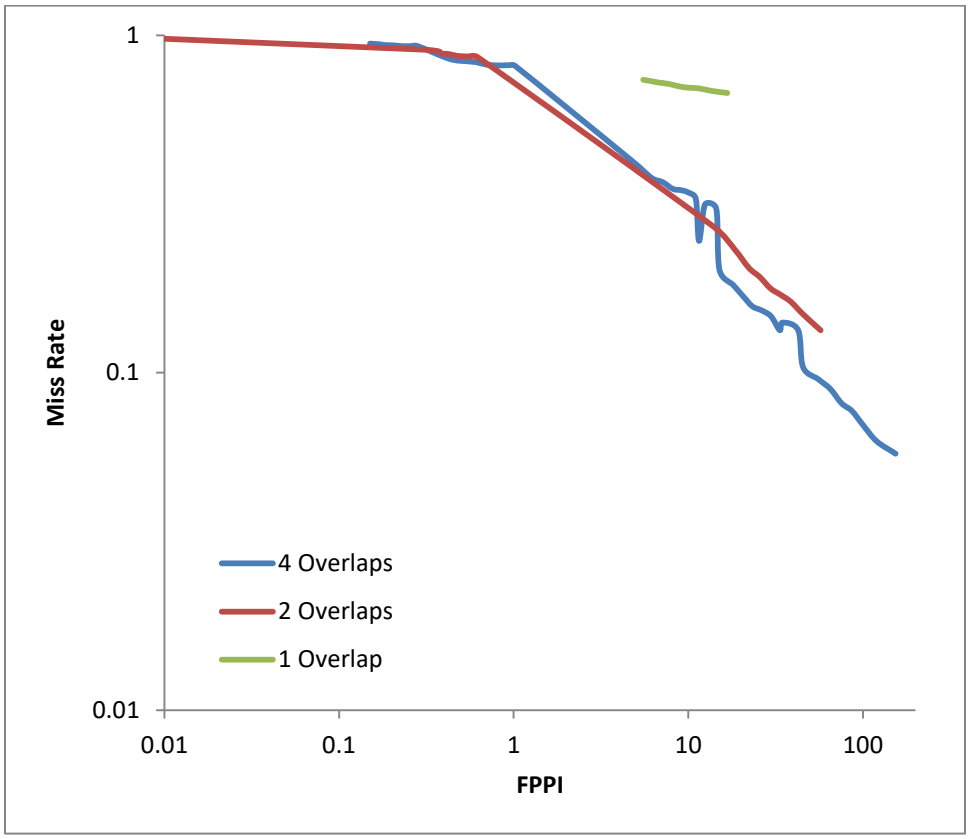


Figure 25: Supervised System Log-Log Miss Rate vs. FPPI on the First K-fold split

With this in mind, it is easy to see why the false positives occur in Figure 26 which shows the predictions and ground truth locations for four 512x512 input chunks. Yellow boxes show predictions, whereas the pink boxes show ground truth locations.

The first subfigure shows the system finding all ground truth locations, albeit with 3 false positives. It can be seen that each of the three false positives contain a blob which confuses the detector. The second sub figure shows that the system has trouble detecting the darker targets on the road. The third subfigure shows the system correctly rejecting all locations in the scene. Finally, the last subfigure shows that certain textures confuse the detector. In this case the bushes in front of the building are incorrectly detected as

vehicles. It isn't hard to see that the low resolution of the images makes it difficult for the detection network to differentiate between blobs caused by vehicles, and similarly sized blobs caused by other objects.

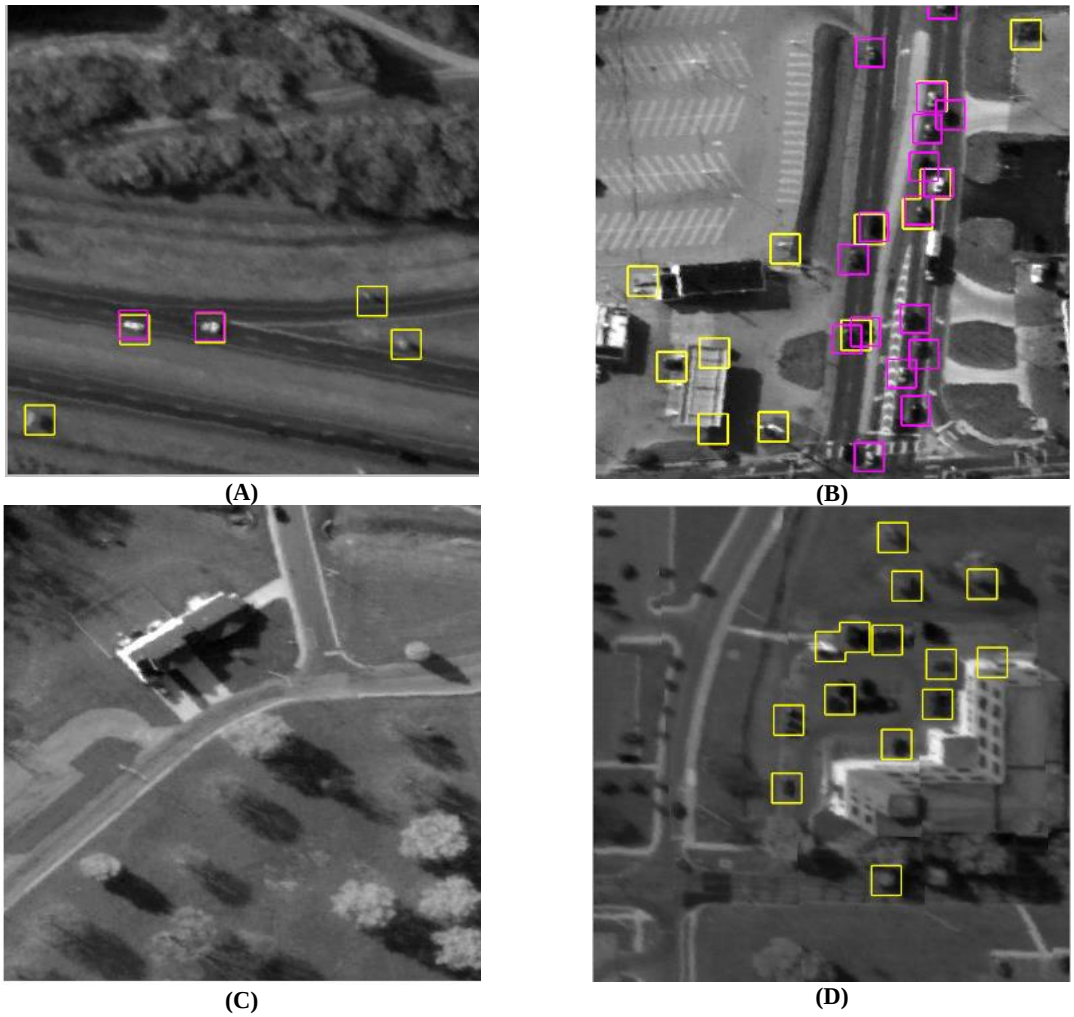


Figure 26: Supervised System Predictions Using Detection Threshold $\alpha = 0.8$
Yellow boxes are predictions, pink boxes are ground truths
(A) All targets are detected, along with three false positive "blobs"
(B) The system has trouble detecting dark vehicles against the road.
(C) The system correctly determines that no targets are present
(D) The system is confused by the bushes in front of the building

Finally the system was run on a second full image from the WPAFB dataset taken roughly 5 minutes after the image used during training. Because no ground truths were provided for this image, only qualitative results were acquired. Figure 27 shows four 512x512 windows with location predictions. As can be seen, the system performs similarly for both images.

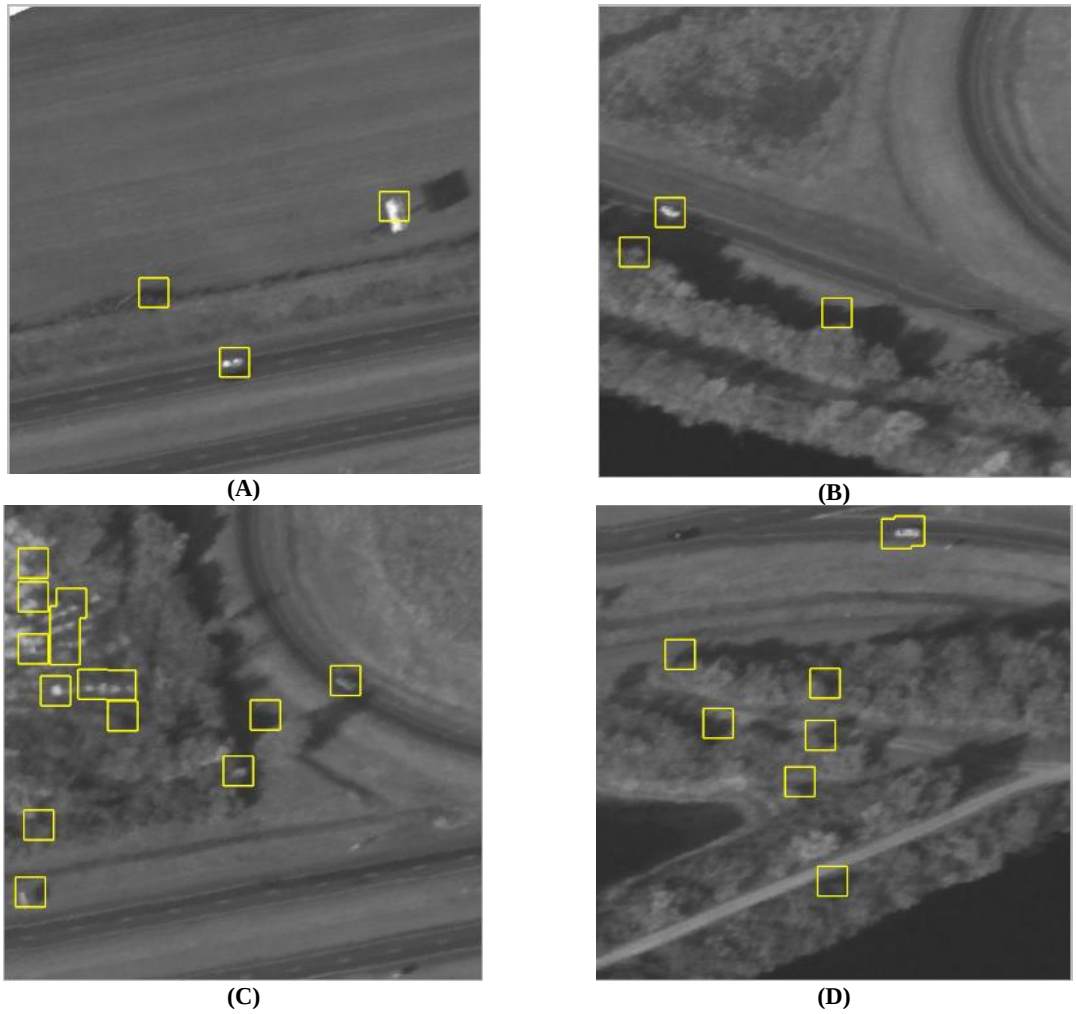


Figure 27: Supervised System Predictions On Second Holdout Image
Yellow boxes are predictions, No Ground truths are provided

5.3 Discussion

5.3.1 Fixed Scale vs. Free Scale

Unlike the semi-supervised approach, for which the fixed scale was an inherent constraint of the system, the supervised methodology fixed the scale simply because doing so resulted in better performance. In theory, a network which leaves all transformation variables as learnable parameters should be better able to model bounding boxes to the found objects. However experimentation with the WPAFB dataset showed that this was a difficult task with the given data. Although an overall cause for this has not been identified there are some likely sources.

First, although the localization target was a fixed size box, there is no discernable difference a mismatch due to a large bounding box, and a mismatch due to a bounding box in the wrong location. As with the semi-supervised approach, a fixed scale may have been avoided by applying a penalty during training which discouraged large bounding boxes. Again however, since the target size is known beforehand, it is simpler to provide the size directly than to tune a penalty parameter.

Another possible solution, if a learnable scale is desired, would be to train the localization task using the bounding boxes directly. For instance, if the localization task was to minimize the IOU between the prediction and ground truth boxes directly, rather than minimizing the reconstruction loss, the inherently account for differences in patch sizes. This would also have the added benefit of evening out the loss associated with high and low contrast patches. As it stands in the current implementation, the loss is dependent on

the contrast of the prediction; low contrast patches will inherently have a lower reconstruction loss because the pixel values are closer together. With direct bounding box regression, the localization loss is independent of visual characteristics.

5.3.2 Low Resolution Targets

With any detection system the main aim is to create an acceptable tradeoff between detecting all of the targets, and rejecting all non-targets. With low resolution images especially, this can be an extremely difficult task. The WPAFB dataset contains very large images, however since the image covers such a large area of land, the large size doesn't translate into high resolution. Even for a human, some of the targets are difficult to identify without enough context information. This can be seen in Figure 26d, where the detector gets confused by the blobs created by the bushes in front of the building. Without an overview of the entire scene, which the detector was not provided, the task of determining what type of object created a low resolution blob is extremely difficult.

5.3.3 System Performance

In terms of localization, the system performs very well. On average, the STN is able to correctly locate nearly 80% of positive samples within a patch with an IOU greater than 0.5. If the match IOU threshold is lowered to 0.3, then the network is able to find nearly 94%. However, with a sliding window approach, the number of windows which don't contain a target far outnumber the number of windows which do. This leads to the need for an extremely strong detector which can reject close to 100% of the negative samples. The huge WAMI scenes found in the WPAFB dataset make this problem difficult on two

fronts. Not only do the relevant targets make up only a miniscule portion of the covered area, but they also have very limited resolution.

The vast majority of windows must be rejected by the detector, while at the same time, a relatively small number of positive samples must be found. With higher resolution targets it may be possible to simply increase the detection threshold, however with the low resolution targets this isn't as easy a task. Even with context it is sometimes hard to distinguish whether a blob is a relevant target, or some other object. As such increasing the detection threshold to the point that irrelevant blobs are rejected tends to make relevant targets be rejected as well.

Chapter 6. Conclusions

6.1 Concluding Remarks

In this thesis, two methodologies for change detection in aerial images were introduced. The basic idea uses a Spatial Transformer network to generate a location prediction, which can then be categorized as containing or not containing a change. A semi-supervised method is presented which is trained to find the area of maximal change in a difference image patch. Because no reference to a target's specific structure is used for training, the semi-supervised system has trouble differentiating relevant changes from background clutter. Therefore a supervised method is also introduced which incorporates a detection network with the Spatial Transformer Network in order to teach the network the structure of the desired targets.

This supervised network performed much better at localizing relevant areas. Even with low resolution targets, this network is able to accurately localize up to 94% of positive targets. However due to the fact that the number of negative samples far outnumbers the amount of positive samples, the detector becomes the limiting factor in the sliding window system. The low resolution of the targets makes it extremely difficult to differentiate between relevant and irrelevant targets which tend to look like simple blobs. This means that even with a high detection threshold, it is likely that false positives will make it through the system. Although the detector is not strong enough to create a

completely autonomous detection system, it does greatly reduce the amount of area which must be covered by human analysts.

The main challenge for both semi-supervised and fully supervised systems seems to be a lack of data. Although both networks are very small compared to current state of the art detectors, they still seem to have unused modeling capability as shown by the learned weights of the supervised network. Although it is possible to shrink the networks, either by retraining a smaller architecture, or by using compression techniques, a more robust solution would be to increase the amount of training data available.

6.2 Future Works

Future improvements to the network can take multiple paths. Different architectures, training methodologies, and data sources are all prime candidates. For instance, a shallower network may be beneficial due to the low resolution of the data. A deep network's consecutive pooling layers will decrease the resolution further still. Even without additional pooling, many of the samples are indistinguishable from simple blobs if no surrounding context is given. This issue may be alleviated by increasing the spatial extent of the filters and input patches, however higher resolution data may still be required for robust deep learning.

The methods used to train the network also have a great impact on its performance. As such it may be beneficial to further experiment with different objective loss functions. One interesting approach would be to train the network to maximize the IOU between its predicted location and the ground truth location. As stated in the supervised discussion,

this would make the localization loss independent from the predicted patch, and allow both high contrast and low contrast predictions to contribute the same amount of error.

The factor with greatest impact on the performance of a neural network is the data used to train it. It may therefore be beneficial to train the network on more scenes from the WPAFB dataset. With a larger dataset, more discrimination can be used when selecting training samples. For instance, training with only high contrast or otherwise discernable targets may decrease overall recall, but it would also increase the precision of the network.

Finally, models such as Digital Imaging and Remote Sensing Image Generation (DIRSIG) may be utilized as an alternative source of image data [45]. This simulation engine developed by the Rochester Institute of Technology (RIT) Digital Imaging and Remote Sensing (DIRS) Laboratory is able to generate radiometrically accurate scenes limited only by the digital models being used. Simulated models are able to produce a large variety of targets, backgrounds, and illumination characteristics, with the additional benefit that accurate ground truths can be generated without the need for hand labeling. Training with simulated data is therefore a logical step towards building a strong aerial detection system which generalizes well to multiple scenarios.

Bibliography

[1] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Comparative study of background subtraction algorithms," *J. Electron. Imaging*, vol. 19, no. 3, Jul. 2010.

[2] K. L. Priddy and D. A. Uppenkamp, "Automated recognition challenges for wide-area motion imagery," in *SPIE*, 2012, vol. 8391.

[3] M. Bryant, Johnson, B. M. Kent, M. Nowak, and S. Rogers, "'Layered Sensing' Its Definition , Attributes , and Guiding Principles for AFRL Strategic Technology Development," *Case # WPAFB-08-1245,*. 2008.

[4] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," *Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 2017–2025, 2015.

[5] S. K. Sønderby, C. K. Sønderby, L. Maaløe, and O. Winther, "Recurrent Spatial Transformer Networks," *CoRR*, Sep. 2015.

[6] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 1–8.

[7] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004, vol. 2, p. 28–31 Vol.2.

[8] S. Varadarajan, P. Miller, and H. Zhou, "Spatial mixture of Gaussians for dynamic background modelling," in *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2013*, 2013, pp. 63–68.

[9] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Static and moving object detection using flux tensor with split gaussian models," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 420–424.

[10] Y. Chen, J. Wang, and H. Lu, "Learning sharable models for robust background subtraction," in *IEEE International Conference on Multimedia and Expo (ICME)*, 2015, pp. 1–6.

[11] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, "A self-adjusting approach to change detection based on background word consensus," in *Proceedings - 2015*

IEEE Winter Conference on Applications of Computer Vision, WACV 2015, 2015, pp. 990–997.

[12] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, "SuBSENSE: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, Jan. 2015.

[13] S. Bianco, G. Ciocca, and R. Schettini, "How Far Can You Get By Combining Change Detection Algorithms?," *CoRR*, May 2015.

[14] A. Rosenfeld, "Automatic Detection of Changes in Reconnaissance Data," in *Proc. 5th Conv. Mil. Electron.*, 1961, pp. 492–499.

[15] K. E. Price and R. Reddy, "Change detection and analysis in multi-spectral images," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977, vol. 2, pp. 619–625.

[16] K. Price, "Symbolic Matching of Images and Scene Models," *Proc. Work. Comput. Vis.*, pp. 105–112, 1982.

[17] B. G. Lee, V. T. Tom, and M. J. Carlotto, "A Signal-Symbol Approach to Change Detection," in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, 1986, pp. 1138–1142.

[18] Z. Jianqing, Z. Zuxun, F. Zhen, and F. Hong, "Change detection from aerial images acquired in different durations," *Geo-spatial Inf. Sci.*, vol. 2, no. 1, pp. 16–20, Jan. 1999.

[19] M. J. Carlotto, "Detection and analysis of change in remotely sensed imagery with application to wide area surveillance," *IEEE Trans. Image Process.*, vol. 6, no. 1, pp. 189–202, 1997.

[20] X. Liu and R. G. Lathrop, "Urban change detection based on an artificial neural network," *Int. J. Remote Sens.*, vol. 23, no. 12, pp. 2513–2518, Jan. 2002.

[21] C. Clifton, "Change Detection in Overhead Imagery Using Neural Networks," *Appl. Intell.*, vol. 18, no. 2, pp. 215–234, 2003.

[22] A. N. Gorban, B. Kegl, D. C. Wunsch, and A. Zinovyev, *Principal manifolds for data visualization and dimension reduction*. Springer, 2007.

[23] Y. Gweon and Y. Zhang, "Change detection for aerial photo database update," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 5, no. 2, pp. 927–932, 2008.

[24] S. Ghosh, S. Patra, and A. Ghosh, "A Neural Approach to Unsupervised Change Detection of Remote-Sensing Images," in *Studies in Computational Intelligence*, vol. 83, Springer Berlin Heidelberg, 2008, pp. 243–268.

[25] P. Sidike, A. Essa, F. Albalooshi, V. Asari, and V. Santhaseelan, “Automatic building change detection in wide area surveillance,” in *2015 National Aerospace and Electronics Conference (NAECON)*, 2015, pp. 54–57.

[26] N. Bourdis, D. Marraud, and H. Sahbi, “Constrained optical flow for aerial image change detection,” in *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2011, pp. 4176–4179.

[27] J. Zollweg, A. Schlamm, D. B. Gillis, and D. Messinger, “Change detection using mean-shift and outlier-distance metrics.”

[28] J. A. Albano, D. W. Messinger, A. Schlamm, and W. Basener, “Graph Theoretic Metrics for Spectral Imagery with Application to Change Detection.”

[29] M. Gong, J. Zhao, J. Liu, Q. Miao, and L. Jiao, “Change Detection in Synthetic Aperture Radar Images Based on Deep Neural Networks,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 1, pp. 125–138, Jan. 2016.

[30] Z. H. Sun, M. Leotta, A. Hoogs, R. Blue, R. Neuroth, J. Vasquez, A. Perera, M. Turek, and E. Blasch, “Vehicle Change Detection from Aerial Imagery using Detection Response Maps.”

[31] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Computation*, vol. 1, no. 4. MIT Press, pp. 541–551, Dec-1989.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning ICML 09*, 2009, vol. 2008, pp. 1–8.

[35] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” Master’s Thesis, Department of Computer Science, University of Toronto, 2009.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[37] A. Coates, H. Lee, and A. Y. Ng, “An Analysis of Single-Layer Networks in Unsupervised Feature Learning,” in *AISTATS*, 2011, vol. 14.

[38] P. Baldi and K. Hornik, “Neural networks and principal component analysis: Learning from examples without local minima,” *Neural Networks*, vol. 2, no. 1, pp. 53–58, Jan. 1989.

[39] H. L. Y. B. and P. A. M. Vincent, “Extracting and Composing Robust Features with Denoising Autoencoders,” in *Proceedings of the Twenty-fifth International Conference on Machine Learning*, 2008, pp. 1096–1103.

[40] Daniel Nouri, “Nolearn.” https://github.com/dnouri/nolearn, Github.

[41] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, E. Battenberg, and A. van den Oord, “Lasagne: First release.” 2015.

[42] J. Bergstra, O. Breuleux, F. F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: a CPU and GPU math compiler in Python,” *Proc. Python Sci. Comput. Conf.*, no. Scipy, pp. 1–7, 2010.

[43] M. Everingham, L. Van Gool, C. K. I Williams, J. Winn, A. Zisserman, M. Everingham, L. K. Van Gool Leuven, B. CKI Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” *Int J Comput Vis*, vol. 88, pp. 303–338, 2010.

[44] U.S. Air Force Research Labs, “SDMS: WPAFB 2009 Dataset,” *https://www.sdms.afrl.af.mil/index.php?collection=wpafb2009*. 2009.

[45] A. A. Goodenough and S. D. Brown, “DIRSIG 5: Core design and implementation,” in *SPIE*, 2012, vol. 8390.

Appendix I – Evaluation Equations

TP, TN:	True Positive / Negative,
FP, FN:	False Positive / Negative
Precision (Pr):	$\frac{TP}{TP+FP}$
Recall (Re):	$\frac{TP}{TP+FN}$
Specificity (Sp):	$\frac{TN}{TN+FP}$
False Pos. Rate (FPR):	$\frac{FP}{FP+TN}$
True Pos. Rate (TPR):	See Recall
Miss Rate:	$\frac{FN}{FN+TP}$
F-Measure:	$\frac{2\cdot \mathrm{Pr}\cdot Re}{\mathrm{Pr}+Re}$
Percentage Wrong Class (PWC):	$\frac{FN+FP}{TP+FN+FP+TN}$